FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY UNIVERSITY OF MALAYA

IMPLEMENTATION OF THREE COLOUR MARKER IN TRAFFIC CONDITIONER FOR DIFFSERV NETWORK SIMULATOR

Lim Lee Wen
(WEK 010141)

Under the Supervision of
Mr. Phang Keat Keong

Moderator
Mr. Ang Tan Fong

SESSION 2003/2004

This project is submitted to the Faculty of Science and Information Technology,
University of Malaya,
in partial fulfillment of the requirement of the Bachelor of Computer Science

ABSTRACT

The Differentiated Services (DiffServ) is architecture has been proposed to achieve scalability by aggregating traffic classification state, which is conveyed by means of IP-layer packet marking using the DS field. In order to ensure that the QoS requirements for each customer are met, the incoming traffic will be separate into different classes to provide a better level of service to all those packets that lie within the traffic profiles, which contracted between customers and internet service providers.

The network simulator is a tool to analyze and study the behavior of DiffServ networks without the existing of real network. The simulation of the simulator will show the results that initiate the real network environment.

The main purpose of this project is to implement the three colour marker for traffic conditioning in the DiffServ network simulator. In this project, there are two types of three colour marker which user is allowed to choose either one for the network simulator. The three colour marker is to govern how traffic is marked and conditioned upon entry to a DiffServ network.

Finally, object oriented approach will be used for the development of the three colour marker in traffic conditioner by using Java programming language.

ACKNOWLEDGEMENT

One of the greatest pleasures of writing this report is acknowledging the efforts of many people whose hard work, cooperation, friendship, and understanding were crucial throughout the undergoing of this project.

Firstly, I would like to thank Mr. Phang Keat Keong, my supervisor for his guidance and advice throughout the entire project. The time that he has shared with me has made a great contribution to the success of the project. Following that, I would like to thank my considerate and kind moderator, Mr. Ang Tan Fong for his valuable pointers and comment on this project.

In addition, I would like to express my gratitude to my discussion group members, Ms. Tang Geck Hiang, Ms. Malini Subramaniam, Mr. Chia Kai Yan, Mr. Andrew Chiam Ming Jer, Mr. Chee Wai Hong, Mr. Chan Chin We and Mr. Au Yee Boon for their willingness to share ideas and informations during discussions session to make this project successful. Next, I would like to thank the master students, Mr. Ng Eng Seong, Mr Soo Wing King and Mr. Tay Lik Czek for sharing their knowledge and experience.

Finally yet importantly, I would like to express my appreciation to my course mates and family members for their encouragement and patience in the succession of this project.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 Introduction to Network Technologies

The growth of the Internet over the last several years has placed a tremendous strain on the besieged superhighway: in the continual increase in users, connection speeds, backbone traffic, Internet Service Provider (ISP) networks and new applications.

Nowadays Internet is a worldwide commercial data network. It has been the proving ground for commerce, managed public data services including intranets and a broadcast medium, all of which presuppose a level of service that includes dependability, predictability and consistent performance. As a result, carriers and Internet providers today face the challenge to scale the capacity, performance and predictability of their network infrastructure and to offer enhanced data services to support customers' TCP/IP applications and emerging markets.

Service providers are looking for ways to offer additional services. As their networks grow along with their customer base, their key concerns are to scale service offerings, offer new services and manage their networks for optimal performance. MPLS technology enables Internet Service Providers to offer additional services for their customers, scale their current offerings, and exercise more control over their growing networks. Traditionally, ISPs offered the same level of performance to all of their customers, which is called the Best-Effort service. The Internet attempts to route all

packets as soon as possible but provides no guarantees regarding the Quality of Service provided to a user. QoS refers to the level of the various properties of a traffic stream e.g. throughput, delay, loss, jitter etc. In the best effort service model, all traffic are treated equally by the routers, and congestion at a router forces all traffic streams passing through that router to slow down. Most differentiation among customers has been probably only in the connectivity type.

In recent years though, ISPs increasingly demand new ways of service differentiation, because of new applications emerged, which require other service qualities. Moreover, business users won't use the Internet to transfer their strategically important information if it cannot assure a required quality of service (QoS). Differentiated Services (DiffServ) architecture delivers this distinctive service functionality for network service providers. Thus, the combination of DiffServ and MPLS seems to be a very attractive strategy for backbone network providers.

1.2 Introduction to Network Simulator

The Internet's rapid growth has spurred development of new protocols and algorithms to meet changing operational requirements—such as security, multicast transport, mobile networking, policy management, and quality-of-service support. Development and evaluation of these operational tools requires answering many design questions.

Network simulators provide a rich opportunity for efficient experimentation. Disparate research efforts using a common simulation environment can yield substantial benefits, including

- improved validation of the behavior of existing protocols,
- a rich infrastructure for developing new protocols,
- the opportunity to study large-scale protocol interaction in a controlled environment,
- easier comparison of results across research efforts,
- Test internet protocols under varied conditions to determine whether they are robust and reliable.

1.3 Project Objective

Traffic conditioning is an essential part of DiffServ Network, which affects the overall performance of the network. Proper traffic management helps in ensuring efficient and fairness operation in the network.

The main objective of this project is to study and understand on the diffserv network especially in traffic conditioning. This including the researching work and discussion about the diffserv network including the network technologies before diffserv is proposed. Next, the second objective of this project is to implement the traffic

conditioning in the UMJanetsim. The needs of study and analyst the existing umjanaetsim simulator is needed before the development stage start.

1.4 *Project Scope*

This project include the study of traffic conditioning in DiffServ network and implement the three colour marker of traffic conditioning in the edge router of DiffServ Network. It started with a study on DiffServ network. Besides, it is also include the network architecture and the behavior of the DiffServ network.

Other than that, the project scope also cover the analysis on the existing network simulators. It includes the study on the advantages and disadvantages of these simulators. This is to analyze the problems of existing network simulators and the minimum requirement of a network simulator.

The review includes the simulator needed for a DiffServ network, such as DiffServ network, MPLS network, TCP/IP and various type of network simulator in order to get the idea of how to build the DiffServ simulator.

This project will develop a DiffServ network simulator with following features:

- A user friendly graphical user interface (GUI).

- Implementation of three colour marker to indicate, mark and simulate the various condition of the packet that transverse in the network
- There are tow types of three colour marker, user is allow to choose either one according to their needs.

1.5 *Goal of Project*

The main goal of the project is to apply traffic conditioning of single rate Three Colour Marker and two rates Three Colour Marker for traffic conditioning in Umjanetsim. The marker in traffic conditioning is responsible to mark the packets. The packets will be monitor by the metering to confirm the packets either to be send or to be drop. The dropper will determine which packets to be drop by recognize the colour that has been mark on the packets.

In three colour marker, the packets will be mark as green, yellow and red. There are two types of three colour marker: single rate Three Colour Marker and two rates Three Colour Marker. Each of the marker will use different types of algorithm. The algorithm and the implementation of both marker will be explain in more details in the following chapters.

1.6 Project Expected Outcome

The implementation of single rate three colour marker and two rate three colour marker will affect the performance of the existing network simulator. After the implementation, the traffic stream will be monitor and control by the traffic conditioner. Those packets, which are violating the service level agreement, will be discarded else, it will be transmitting from the source node to destination node. The implementation would not affect the existing features and components in the existing network simulator. The GUI of the simulator will remain the same.

1.7 Project Schedule

The project schedule is shown in Table 1.1:

	Task Name	Duration	Start	Finish
1	Project Definitions	10 days	Mon 16/06/03	Fri 27/06/03
2	Literature Review	25 days	Wed 25/06/03	Tue 29/07/03
3	Research	30 days	Fri 18/07/03	Thu 28/08/03
4	System Analysis	17 days	Mon 18/08/03	Tue 09/09/03
5	System Design	23 days	Mon 01/09/03	Wed 01/10/03
6	Implementation	70 days	Wed 01/10/03	Tue 06/01/04
7	Testing	25 days	Fri 02/01/04	Thu 05/02/04
8	Documentation	150 days	Fri 25/07/03	Thu 19/02/04

Table 1. 1 Project Schedule

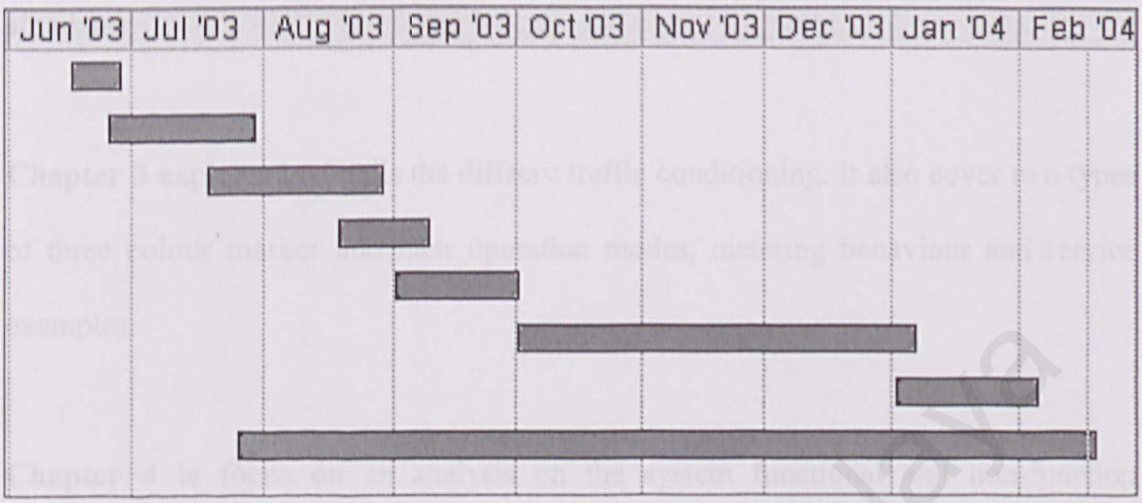


Figure 1 1 Project Gant Chart

1.8 Report Layout

This report has been divided to 8 chapters, which are organized as follows:

Chapter 1 introduces the current networking technologies associates with this project and UMJanetSim, project definition, project objectives, goals of projects, project expected outcome, project schedule and report layout.

Chapter 2 is about literature review, which covers the literature review and research work done during the project. The chapter mainly covers 3 sections. The first section is a review of networking technologies that is ATM and LAN emulation. The second section covers the evaluation of current existing network simulator and the last

section reviews the programming approaches that can be used to develop the simulator.

Chapter 3 explains in details the diffserv traffic conditioning. It also cover two types of three colour marker and their operation modes, metering behaviour and service examples.

Chapter 4 is focus on an analysis on the system functional and non-function requirements. It included the problem solving techniques that decomposes a system into tits component pieces for the purposes of studying how well those component parts work and interact to accomplish their purpose.

Chapter 5 discuss about the design of the UMJanetsim network simulator. Beside that, the design of the new component which is three colour marker is also included. **Chapter 6** is present the implementation of srTCM and trTCM in UMJanetSim netowkr simulator. Through out this user, the way to implement the three colour marker and the Java coding is displayed and explained in details.

Chapter 7 is describe the testing phase in the development of three colour marker. The testing phase is divided in three category which are component testing, module testing and system testing. Each level of testing's results are shown in this chapter.

Chapter 8 is the final part of this report. This chapter is to conclude the whole project after the system is successfully been developed where it has achieve the project objective requirements and objectives.

CHAPTER 2 Literature Review

2.1 TCP/IP

2.1.1 Introduction

TCP/IP is the Transmission Control and Internet Protocol developed by the Dept of Defense's Advanced Projects Research Agency (ARPA) in 1969. TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. It is a set of protocols, or a protocol suite, that defines how all transmissions are exchanged across internet. It can also be used as a communications protocol in a private network (either an intranet or an extranet).

TCP/IP is a two-layer program. The higher layer is Transmission Control Protocol (TCP), and the lower layer is Internet Protocol (IP). TCP/IP uses the client/server model of communication in which a computer user (a client) requests and is provided a service (such as sending a Web page) by another computer (a server) in the network. TCP/IP communication is primarily point-to-point, meaning each communication is from one point (or host computer) in the network to another point or host computer. TCP/IP and the higher-level applications that use it are collectively said to be "stateless" because each client request is considered a new request unrelated to any previous one (unlike ordinary phone conversations that require a dedicated

connection for the call duration). Being stateless frees network paths so that everyone can use them continuously. (Note that the TCP layer itself is not stateless as far as any one message is concerned. Its connection remains in place until all packets in a message have been received.) *(Forouzan, 2000)*

2.1.2 TCP

TCP manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. It enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent.

2.1.3 IP

IP handles the address part of each packet so that it gets to the right destination. It is something like the postal system. It allows you to address a package and drop it in the system, but there is no direct link between you and the recipient. TCP/IP Each gateway computer on the network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, they will be reassembled at the destination. There are two version of IP, which are Ipv4 and IPv6 (under development).

2.1.4 TCP/IP & OSI

TCP was developed before the OSI model. The TCP/IP protocol is made of five layers: physical, data link, network, transport, and application. (Sometimes is TCP/IP is defined made of four layers which data link layer and network layer is combined as internet layer.). The application layer in TCP/IP can be equated with the combination of session, presentation and application layer of the OSI model.

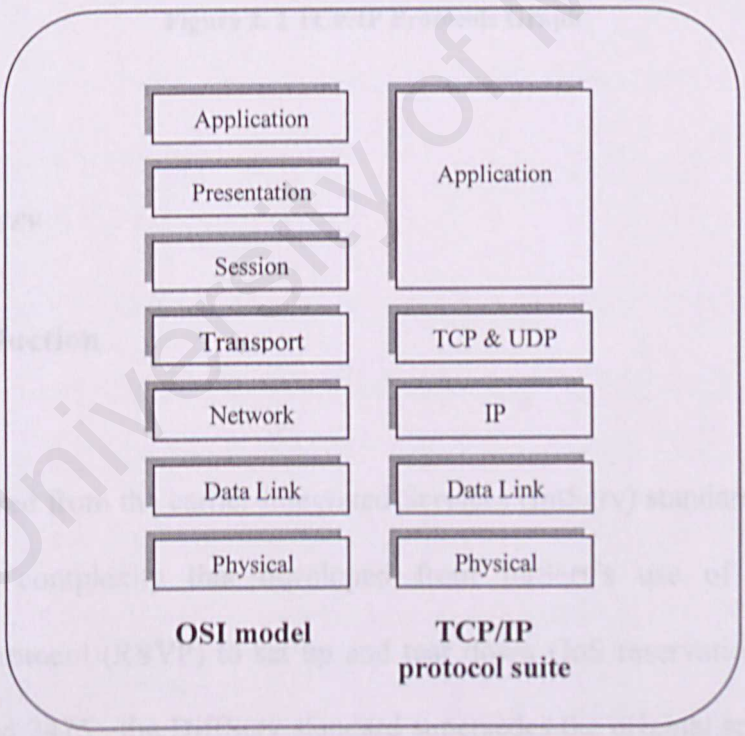


Figure 2. 1 OSI model and TCP/IP protocol suite

TCP/IP Protocols Graph

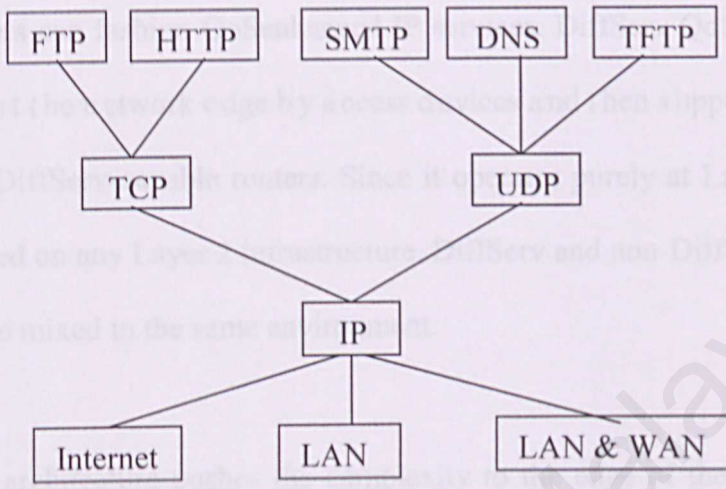


Figure 2. 2 TCP/IP Protocols Graph

2.2 DiffServ

2.2.1 Introduction

DiffServ evolved from the earlier Integrated Services (IntServ) standard. It eliminates much of the complexity that developed from IntServ's use of the Resource Reservation Protocol (RSVP) to set up and tear down QoS reservations. Defined in RFCs 2474 and 2475 , the DiffServ standard supersedes the original specification for defining packet priority described in RFC 791 . DiffServ increases the number of definable priority levels by reallocating bits of an IP packet for priority marking.

DiffServ is a Layer 3 solution that addresses QoS requirements in a connectionless environment. Its main purpose is to standardize a set of QoS building blocks with which providers can fashion QoSenhanced IP services. DiffServ QoS is meant to be implemented at the network edge by access devices and then supported across the backbone by DiffServ-capable routers. Since it operates purely at Layer 3, DiffServ can be deployed on any Layer 2 infrastructure. DiffServ and non-DiffServ routers and services can be mixed in the same environment.

The DiffServ architecture pushes the complexity to the edge of the network where there are fewer flows in parallel. Traffic classification and packet filtering is performed here. Micro flows are aggregated into traffic classes, solving the scalability problem in the core of the network.

DiffServ provides scalable and "better than best-effort" QoS. A DiffServ router is stateless and do not keep track of individual microflows, making it scalable to be deployed in the Internet. The DiffServ Code Point (DSCP) in the Differentiated Services (DS) field of the IP header identifies the Per Hop Behavior (PHB) associated with the packet, which is used to specify queuing, scheduling, and drop precedence. There are three defined PHBs:

i. Best effort,

ii. Assured Forwarding (AF), and

iii. Expedited Forwarding (EF)

At the ingress node in a DiffServ domain, the DSCP value is determined based on multifield classification of the incoming packet. At the interior nodes, the PHB is determined from the DSCP and appropriate QoS treatment is applied to the packet. At the egress node, the packet is routed to the next hop in the next domain. Traffic conditioning is performed at the boundary nodes to ensure the traffic streams conform to the traffic conditioning agreement (TCA) between two domains. *(CISCO, 2001)* (*Dr. Ljiljana Trajkovic. 2003*)

2.2.2 DSCP

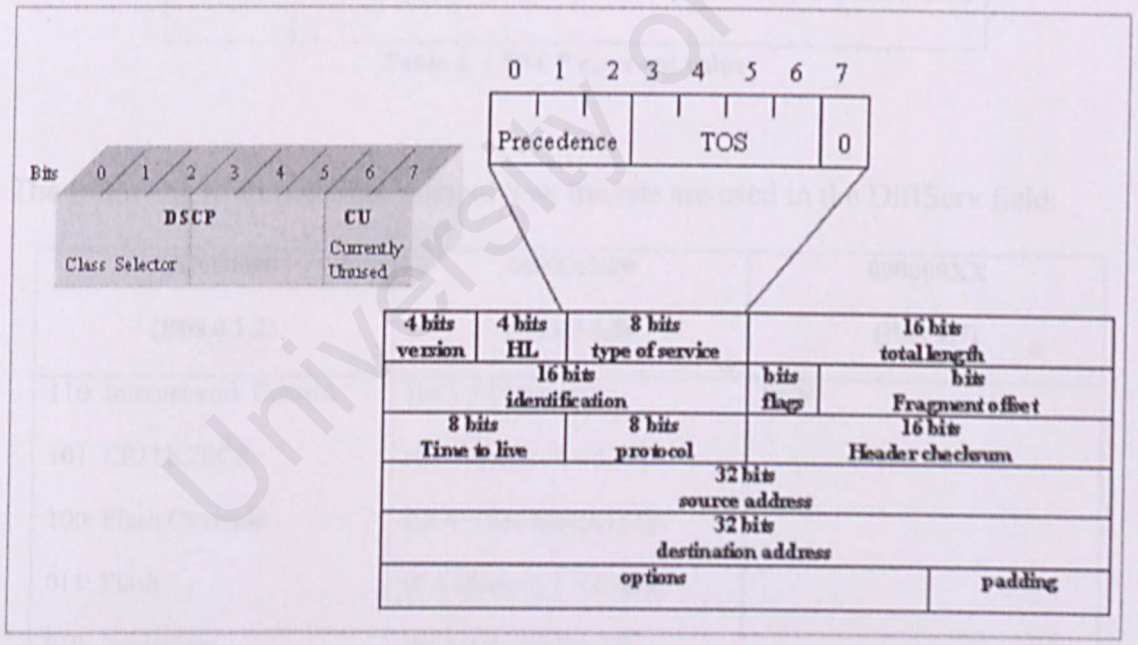


Figure 2. 3 DSCP

In Differentiated Services, DS field [RFC2474] is defined to supersede the IPv4 TOS octet [RFC791] and the IPv6 Traffic Class octet [IPv6] for service level specified purpose. The leftmost 6 bits are used as DSCP; therefore there are 64(2^6) different classes/aggregates available. The other two bits are currently unused and are proposed to be used by Early Congestion Notification (ECN). When determining the per-hop behavior to apply to a received packet, the values of the CU bits are ignored by differentiated services-compliant nodes,. (*Lynda Linney. 1999*)

xxxxx0	reserved for standards
000000	default PHB Codepoint
xxx000	Class Selector Codepoints: reserved for backwards

Table 2. 1 DSCP reserved value

The following is an illustration about how the bits are used in the DiffServ field:

XXX00000 **(Bits 0,1,2)**	**000XXX00** **(Bits 3,4,5)**	**000000XX** **(Bits 6,7)**
110: Internetwork Control 101: CRITIC/ECP 100: Flash Override 011: Flash 010: Immediate 001: Priority	Bit 3 = Delay [D] (0 = Normal; 1 = Low) Bit 4 = Throughput [T] (0 = Normal; 1 = High) Bit 5 = Reliability [R] (0 = Normal; 1 = High)	ECN

Table 2. 2 DSCP bit

The standardized DiffServ field of the packet is marked with a value so that the packet receives a particular forwarding treatment or PHB, at each network node. The default DSCP is 000 000 and class selector DSCPs are backward compatible with IP precedence.

When converting between IP precedence and DSCP, leave the upper three bits as 0 and then match the lower three bits. In other words:

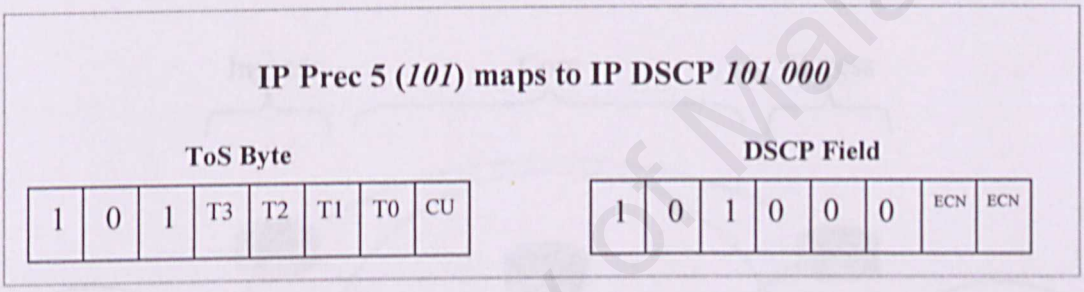


Figure 2. 4 ToS byte vs DSCP field

The DiffServ standard utilizes the same precedence bits (the most significant bits: 0, 1, and 2) for priority setting, but further clarifies the definitions, offering finer granularity through the use of the next three bits in the ToS field. DiffServ reorganizes and renames the precedence levels (still defined by the three most significant bits of the ToS field) into the following categories:

Precedence 7	Stays the same (link layer and routing protocol keep alive)
Precedence 6	Stays the same (used for IP routing protocols)
Precedence 5	Express Forwarding (EF)
Precedence 4	Class 4

Precedence 3	Class 3
Precedence 2	Class 2
Precedence 1	Class 1
Precedence 0	Best effort

Table 2. 3 Tos Field Categories

2.2.3 DiffServ Architecture

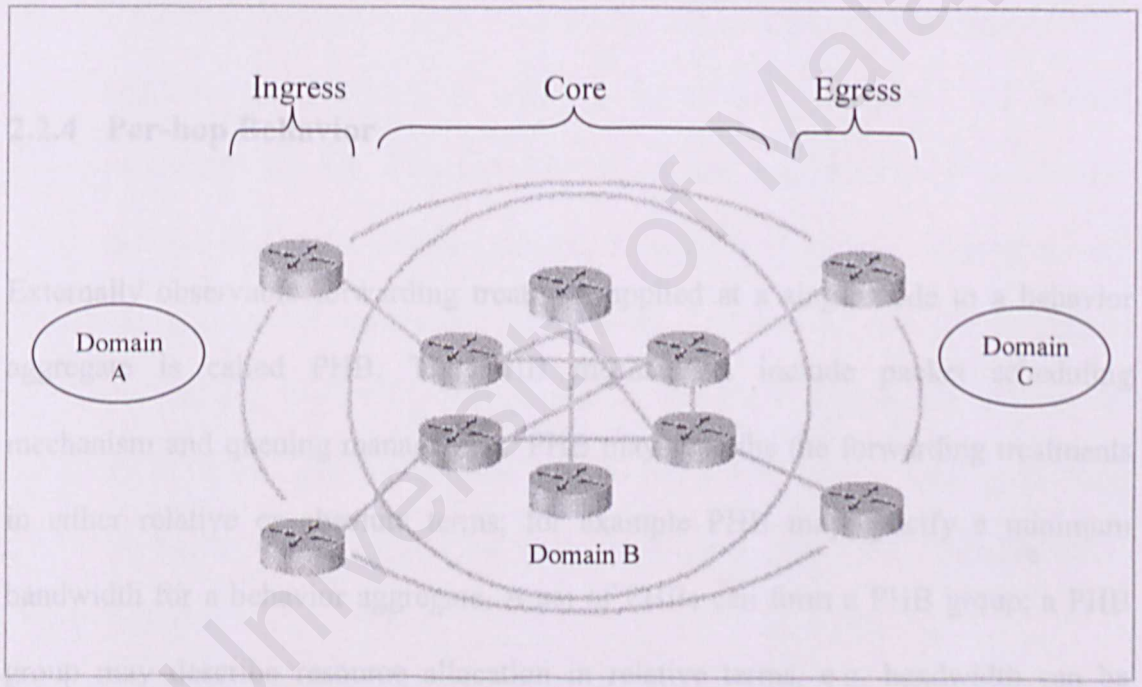


Figure 2. 5 DiffServ Architecture

A DS-Domain is made up of DS Edge/Boundary nodes (ingress & egress nodes) and DS Core nodes. Typically, the DS Boundary node performs traffic conditioning. A traffic conditioner typically classifies the incoming packets into pre-defined

aggregates, meters them to determine compliance to traffic parameters (and determines if the packet is in profile, or out of profile), marks them appropriately by writing/re-writing the DSCP, and finally shapes (buffers to achieve a target flow rate) or drops the packet in case of congestion. Figure 5 illustrates the typical traffic conditioner at the edge of a DS-Domain. A DS Internal node enforces the appropriate PHB by employing policing or shaping techniques, and sometimes re-marking out of profile packets, depending on the policy or the SLA. *(Cisco Systems. 2001)*

2.2.4 Per-hop Behavior

Externally observable forwarding treatment applied at a single node to a behavior aggregate is called PHB. The PHB mechanisms include packet scheduling mechanism and queuing management. PHB may describe the forwarding treatments in either relative or absolute terms; for example PHB may specify a minimum bandwidth for a behavior aggregate. A set of PHBs can form a PHB group; a PHB group may describe resource allocation in relative terms, e.g. bandwidth can be allocated in terms of drop priority. There are four standards of PHBs:

i. *The Default PHB (RFC2474)* – The default PHB essentially specifies that a packet marked with a DSCP value (recommended) of '000000' gets the traditional best effort service from a DS-compliant node (a network node that complies to the entire core DiffServ requirements). Also, if a packet arrives at

a DS-compliant node and its DSCP value is not mapped to any of the other PHBs, it will get mapped to the default PHB.

ii. *Class-Selector PHB (RFC2474)* – To preserve backward compatibility with the IP-Precedence scheme, DSCP values of the form 'xxx000', where x is either 0 or 1 are defined. These Codepoints are called Class-Selector Codepoints. These PHBs ensure that DS-compliant nodes can co-exist with IP-Precedence aware nodes, with the exception of the DTS bits.

iii. *Expedited Forwarding (RFC3246/2598)* – the EF PHB is the key ingredient in DiffServ for providing a low-loss, low-latency, low-jitter, and assured bandwidth service. Although EF PHB when implemented in a DiffServ network provides a premium service, it should be specifically targeted toward the most critical applications, since if congestion exists, it is not possible to treat all or most traffic as high priority. EF PHB is especially suitable for applications (like VoIP) that require very low packet loss, guaranteed bandwidth, low delay, and low jitter. The recommended DSCP value for EF is '101110'.

Per-hop Behavior Group: a set of one or more PHBs that can only be meaningfully specified and implemented simultaneously, due to a common constraint applying to all PHBs in the set such as a queue servicing or queue management policy.

Assured Forwarding *(RFC2597 (AFxy) PHB)* – It defines a method by which Behavior Aggregates can be given different forwarding assurances. AF is separated to four classes; each class has a specific set of resources (buffers/bandwidth) and its own queue. E.g. Gold, Silver, Bronze service

y\x	Class 1	Class 2	Class 3	Class 4
1. Low Drop	001010	010010	011010	100010
2. Medium Drop	001100	010100	011100	100100
3. High Drop	001110	010110	011110	100110

Table 2. 4 Assured Forwarding classes

2.2.5 Comparison of DiffServ with other QoS approaches

Two other well known approaches to provide QoS over the Internet are MPLS and Integrated Service. MPLS is based on circuit switching but is similar in some respects to DiffServ. In a MPLS a label is prefixed to a packet and this label is used at a router to select the path on which the packet is to forwarded, the new label of the packet and the per hop behavior applied to the packet at the router. As the number of labels can be large and labels have only local significance, the possible number of behavior aggregates that can be supported by MPLS is potentially much larger than DiffServ. MPLS also provides the option of selecting the route taken by a packet based on the requirements of the user. On the downside, MPLS requires end-to-end signaling before packets can be sent over a virtual circuit. The amount of state that needs to be

maintained at the routers is larger than that required for DiffServ and MPLS also suffers from the disadvantages of circuit switching.

Integrated Services architecture aims to provide QoS guarantees by reserving resources at all nodes that lie on the path of a traffic stream. RSVP is used as the signaling protocol for Integrated Services. However for high speed links, the number of traffic streams for which state needs to be maintained can be very large. This can also have a detrimental effect on the maximum rate at which a router can forward packets, as a large number of entries would have to be searched to get the information about the resources reserved for a particular traffic stream. Like MPLS, resources need to be reserved along the path from the sender to the receiver before any traffic can be sent.

2.3 MPLS

MPLS is an IETF initiative that integrates Layer 2 information about network links into Layer 3 (IP). The main goal of MPLS was to bring the speed of Layer 2 switching to Layer 3. MPLS provides connection oriented switching and allow routers to make forwarding decisions based on the contents of a simple label, rather than by performing a complex route lookup based on destination IP address. This initial justification for technologies such as MPLS is no longer perceived as the main

benefit, since Layer 3 switches (ASIC-based routers) are able to perform route lookups at sufficient speeds to support most interface types. (*Jerry Ryan. 1998*)

MPLS provides a feasible way of implementing traffic engineering with different granularity levels. Explicit routing can be performed without carrying the completely explicit route in every single IP packet transported. Label edge router (LER) wills assigns a label/identifier for the each incoming packets based on a forwarding equivalence class (FEC). These labels not only contain information based on the routing table entry (i.e., destination, bandwidth, delay, and other metrics), but also refer to the IP header field (source IP address), Layer 4 socket number information, and differentiated service. Once the packets are labeled, it will be forward along a label switch path (LSP). All the LSR have just to perform a label look up, swap the label to one meaningful to the next hop and decrement, if applicable, the TTL (Time To Live) of the packet.

MPLS is destined to provide a new technical foundation for the next generation of multi-user, multiservice internetworks. The promise is for higher performance, another order of magnitude increase in scalability, improved and expanded functionality, and the flexibility to match the user's quality of service requirements more closely.

2.3.1 MPLS Label

MPLS is not tied to any concrete protocol; it uses different label encapsulation depending on the link layer below. In ATM, for instance, MPLS labels are the VPI (Virtual Path Identifier) and the VCI (Virtual Channel Identifier) in the ATM headers. When the link layer does not provide a means for encapsulating MPLS labels, shim header is used. It is added after the link layer header and its format is shown in the figure below. It is a 32-bit, locally significant identifier used to identify a FEC.

Label (20 bits) – A locally significant ID used to represent a particular FEC during the forwarding process.
Exp (3 bits) – Previously called class of service (CoS), it is now considered an experimental range. Currently, this field is being considered for QoS implementations.
S (1 bit) – Used to signify if label stack is present. If the label is the only one present or at the bottom of the stack, the bit will be a value of zero.
TTL (8 bits) – Field used to signify the number of MPLS nodes that a packet has traversed to reach its destination. The value is copied from the packet header and copied back to the IP packet header when it merges from the LSP.

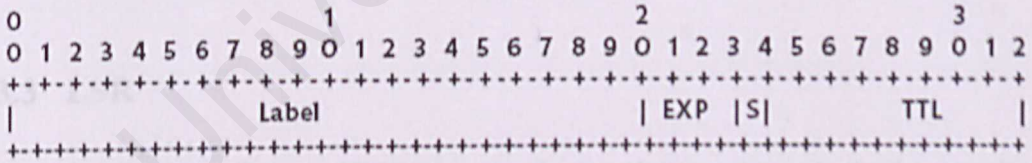


Figure 2. 6 Shim Header

The shim header provides a TTL (Time to Live) field. When an LSR swaps a label, encapsulated using the shim header into a label using ATM encapsulation this information is lost. Multiple shim headers can be carried in a packet in a header stack. This is used when there is a routing hierarchy in the network. Some values of the

labels are reserved. They can indicate that the IP header after the shim header has to be examined or that the routing has to be performed based on a label different than the first one. The shim header has 3 bits reserved for experimental and future use. Some approaches to combine MPLS with DiffServ utilize this experimental field to encapsulate information related to the PHB expected.

2.3.2 LER

The LERs is responsible to classify traffic, apply and remove label to and from data packets. The LER determines whether the traffic is a long lasting flo, implements management policies and access controls, and performs aggregation of traffic into larger flow if it is possible. These are all functions that need to be performed at the boundary between the IP and MPLS worlds. Thus, the capabilities of LERs will be key to the success of an overall label switching environment.

2.3.3 LSR

A Label Switching Router (LSR) is any device that supports both the standard IP control component (i.e.,routing protocols, RSVP, etc.) and a label swapping forwarding component. A label-switching network serves the same purpose as any conventional routed network: it delivers traffic to one or more destinations. The addition of label-based forwarding complements conventional routing but does not replace it. There are three distinctions between LS routing and conventional routing:

2.3.4 LSP

The LSP is essentially the predetermined route that a set of packets bound to a FEC traverse through an MPLS network to reach their destination. Each LSP is unidirectional; therefore, return traffic must use a separate LSP. MPLS provides to options to set up and LSP:

i. Hop-by-hop routing: Each LSR selects the next hop for given FEC.
ii. Explicit routing: The ingress LSR specifies the list of nodes.

2.3.5 LDP

The LDP is a new protocol for the distribution of label binding information to LSRs in an MPLS network. It is used to map FECs to labels, which, in turn, create LSPs. LDP sessions are established between LDP peers in the MPLS network. The peers' exchange the following types of LDP messages:

- Discovery messages: announce and maintain the presence of an LSR in a network
- Session messages: establish, maintain, and terminate sessions between LDP peers
- Advertisement messages: create, change and delete label mapping for FECs

- Notification messages: provide advisory information and signal error information

2.3.6 FEC

The forward equivalence class (FEC) is a representation of a group of packets that share requirements for their transport. All packets in such group are provided the same treatment en route to the destination. As opposed to conventional IP forwarding, in MPLS, the assignment of a particular packet to a particular DEC is done just once, as the packet enters the network. FECs are based on service requirements for a given set of packets or simply from an address prefix. Each LSR builds a table to specify how a packet must be forwarded. This table, called a label information base (LIB), is comprised of FEC-to-label bindings.

2.4 Introduction to Network Simulation

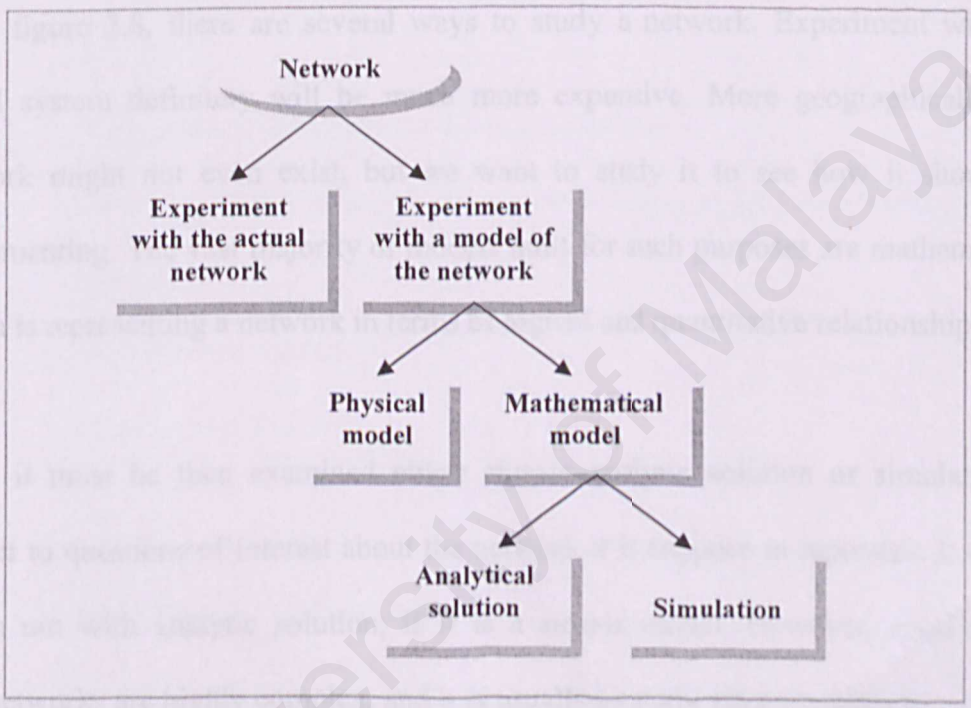


Figure 2. 7 Ways to study network system

In a simulation, we use a computer uses a computer program that duplicates the essential behavior of a real physical system. The purpose of simulator is to evaluate a model numerically, and data are gathered in order to estimate the desired true characteristic of the model. As an example of network simulation, implement a new technology into an internetwork but is not sure if the productivity and performance

would justify the implementing cost. It certainly would be not cost-effective if we try to implement it and then remove it later if it does not work out.

2.4.1 Analytical Solution vs. Simulation

From figure 2.8, there are several ways to study a network. Experiment with the actual system definitely will be much more expensive. More geographically, the network might not even exist, but we want to study it to see how it should be implementing. The vast majority of models built for such purposes are mathematical, which is representing a network in terms of logical and quantitative relationships.

Next, it must be then examined either choose analytic solution or simulation to answer to questions of interest about the network it is suppose to represent. It maybe works out with analytic solution, if it is a simple model. However, most of the internetworks are highly complex, and it is usually to study via a simulation.

2.4.2 Advantages of Simulation

Network simulator is a very useful tool for the network researcher. The advantages of network simulation are as follows:

- Realism - analytical models require more simplification. Simulation can model complex processes.

- Time compression - months of real time can be simulated in seconds on the computer (e.g. weather forecasts)
- Training - simulation requires less mathematical training than analytical modeling
- Presentation of results - results more transparent, especially if system is animated
- Study processes that are intractable analytically
- Observing real process is too expensive, takes too long, or should not be manipulated
- controlled environment
- experimental tools

2.4.3 Disadvantages of Simulation

Everything in this world has their pro and cons. Beside the advantages that listed above, a simulation tool also has its disadvantages. The disadvantages of a simulation tool are listed below:

- correctness (no proof)
- exhaustive search
- usually takes long time
- Failure to optimize - tyranny of large combinations of alternatives
- Long lead times - months of effort might be required to develop a major simulation model

- Lack of generality of results - results only apply to situation in the model and cannot be extended out of context. User must understand the underlying process to start.
- Costs for developing simulation capability - hardware, software, training, support, and staffing
- Model must contain uncertainty - otherwise the solution will be deterministic
- Misuse of simulation - easier to build simulations, thus unqualified people can build incorrect or incomplete models

2.4.4 Discrete-Event Simulation

Discrete-event simulation concerns the modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time. These points in time are the ones at which an event occurs, where an event is defined as an instantaneously occurrence that may change the state of the system. Although discrete-event simulation could conceptually be down by hand calculations, the amount of data that must be stored and manipulated for most real world system.

2.4.5 Study of Various Existing Simulator

Network simulators play a very important role as the emerging speeds and dynamic of the computer network today. There are several types of network simulator in the market. Each of the simulators has different conditions, some are freeware and the others are for commercial purpose. Table 2.6 is showing different type of network simulator with their description.

In following sub topic, all the simulators will be explain in more details with the advantages and disadvantages.

OMNeT++

OMNeT++ is a discrete event simulation environment for simulation of communication networks, but because of its generic and flexible architecture, it is used in other areas like the simulation of complex IT systems, queuing networks or hardware architectures as well. OMNET++'s components are programmed in C++. OMNeT++ has been developed on Linux, but it works just as well on most Unix systems and on Win32 platforms (NT4.0, W2K or XP recommended).

Name	Comments	Conditions	OS
Omnet++	- Discrete-event simulation - Written in C++ - Developed by Andras Varga etc.	Open source. Academic Public Licence.	linux unix NT
Berkeley NS	- UCB/LBNL/VINT Network Simulator - ns (version 2). - Discrete event simulator targeted at networking research. - Provides substantial support for simulation of TCP, routing, and multicast protocols.		
Real 5.0	- Originally intended for studying the dynamic behavior of flow and congestion control schemes in packet-switched data networks - Simulates flow control algorithms such as TCP, and various queueing disciplines - Written in C - Uses NeST simulator	Source is provided. Free of charge, but must fill in and send off a license form.	unix
INSANE	- Internet Simulated ATM Networking Environment - By Bruce Mah. - No longer actively supported. - Designed to test various IP-over-ATM algorithms with realistic traffic loads derived from empirical traffic measurements.	Free, with source.	unix

	- Internet protocols supported include large subsets of IP, TCP, and UDP - Written in C++.'		
NetSim++	- Can be used in areas of communications networks such as: performance measurement for existing or future networks under a wide range of conditions [and] analysis and simulation of queuing systems.	Free.	
opnet		Commercial, but free for university use.	
NIST ATM simulator	- A real-time ATM network emulator. - Their APROPS (PNNI simulator) also seems to be a real-time emulator.	Commercial products.	

Table 2. 5 Network Simulation list

Disadvantages

OMNet++ is lack on an integrated graphical development environment and model library of standard protocols, applications and devices. To write simple modules, the user do not need to learn a new programming language, but they assumed to have some knowledge of C++ programming.

NS2

NS2 is a discrete event simulator targeted at network research. NS2 provides substantial support for simulation of TCP, routing and multicast protocols over wired and wireless networks. NS2 is derived from the REAL network simulator. Now it is supported by DARPA through the VINT project. The simulator is written in C++, and simulation scenarios are designed using the Tcl scripting language.

Advantages

NS2 includes a network emulation interface that permits network traffic to pass between real world network nodes and the simulator. NS2 allows simulation with multiple level of abstraction, where higher abstraction levels trade off accuracy for performance. The simulator measurements do not impact the network by adding extra traffic.

Disadvantages

Although NS2 has a network animation tool that provides network visualization features, but it does not have a GUI for general simulation manipulation and scenario setup.

REAL 5.0

REAL is a network simulator originally intended for studying the dynamic behaviour of flow and congestion control schemes in packet-switched data networks. REAL is written in C, and will run on Digital Unix/ SunOS/ Solaris/ IRIX/ BSD4.3/Ultrix /UMIPS systems on VAX, SUN, SPARC, MIPS, Alpha, SGI or DECstation hardware.

Advantages

The modular design of the system allows new modules to be added to the system with little effort. Source code is provided so that interested users can modify the simulator to their own purposes.

Disadvantages

The user must have strong foundation in C Programming Language in order to change the source code provided to modify the simulator to for their purposes.

INSANE

INSANE is a network simulator designed to test various IP-over-ATM algorithms with realistic traffic loads. It can be used for a variety of studies of application or network behavior in IP internetworks.

INSANE's ATM protocol stack provides real-time guarantees to ATM virtual circuits by using Rate Controlled Static Priority (RCSP) queueing. ATM signalling is performed using a protocol similar to the Real-Time Channel Administration Protocol (RCAP). Internet protocols supported include large subsets of IP, TCP, and UDP. In particular, the simulated TCP implementation performs connection management, slowstart, flow and congestion control, retransmission, and fast retransmits. Various application simulators mimic the behavior of standard Internet applications to provide a realistic workload, including telnet, ftp, WWW, real-time audio and real-time video. INSANE is written in C++ and customization and simulation configuration is performed with Tcl scripts.

Disadvantages

No graphical user interface, only Tk-based graphical simulation monitor provides an easy way to check the progress of multiple running simulation processes.

NetSim++

In a nutshell, NetSim++ is a software package designed to provide a comprehensive work environment for the network modeler. It can be used in areas of communications networks such as performance measurement for existing or future networks under a wide range of conditions [14]. Besides that, it can perform analysis and simulation of queuing systems. NetSim++ is designed specifically for the development and analysis of communications networks. Models can be hierarchically structured, allowing their re-use in different simulations. Specifications are entered graphically with specialized editors. The editors provide an efficient medium for design capture via a consistent set of modern user-interface elements. NetSim++ follows for the hierarchy and communication model a subset of SDL-92 semantics. As with SDL, the active parts are processes; a hybrid approach is used to embed C++ language code with a graphically specified Extended Finite State Machine (EFSM).

Advantage

NetSim++ provides an efficient event-driven Simulation Kernel, a Simulation API and a Base Models Library of components. It takes the design specification and automatically generates an executable simulation. A set of analysis tools is provided to interpret and visualize a large volume of simulation results.

Disadvantage

The current implementation of NetSim++ is available only for UNIX/X Window System platforms.

OPNET

OPNET Modeler is the industry's leading network technology development environment, allowing you to design and study communication networks, devices, protocols, and applications with unmatched flexibility. Modeler is used by the world's most prestigious technology organizations to accelerate the R&D process. Modeler's object-oriented modeling approach and graphical editors mirror the structure of actual networks and network components, so your system intuitively maps to your model. Modeler supports all network types and technologies, allowing you to answer the most difficult questions with confidence.

Advantages

- An intuitive graphical environment that precisely models real networks, devices, protocols, and applications.
- The control over modeling detail needed to support your engineering decisions.
- Built-in support for simulating all types of network technologies.
- The industry's most comprehensive library of standards-based protocol models, with completely open source code .

- An environment designed to model proprietary protocols using Finite State Models, C/C++ and extensive libraries.
- Powerful analysis tools integrated directly into the GUI.
- The industry's most efficient simulation engine, with hybrid simulation capability.
- Outstanding support and services to ensure your success.

Disadvantages

OPNET does not give user an interactive modeling environment with a graphical user interface (GUI) representation capability. It does not provide high-level textual model description. OPNET can only create fixed topologies, which it lacks efficient support for building large and regular topologies. On the other hand, it does not support parallel execution.

NIST ATM/HFC Network Simulator

ATM/HFC Network Simulator was developed at National Institute of Standards and Technology (NIST) to provide a means for researchers and network planners to analyze the behavior of ATM networks without the expense of building a real network.

The simulator is used as a network planning tool by simulating various network configurations and traffic loads, and obtaining statistics such as utilization of network

links and throughput rates of virtual circuits. It also can be used to check circuits' bottlenecks and network congestion.

The simulator can also be used as a protocol analysis tool to study the total system effect of a particular ATM network protocol. Users can investigate the effectiveness of various flow control mechanisms for ATM networks and address such issues as mechanisms for fair bandwidth allocation, protocol overhead, and bandwidth utilization.

NIST has developed the tool using both C language and the X Windows System running on a UNIXTM-based platform.

Advantages

The simulator is designed that modules simulating components of an ATM network can be easily changed, added, or deleted. The NIST ATM Network Simulator gives the user an interactive modeling environment with a graphical user interface.

To execute a simulation, the user creates a network consisting of any number of ATM switches and hosts interconnected by links of variable speeds. Virtual connections between any two hosts are manually established, and applications on hosts are selected and initiated. While a simulation is running, various instantaneous performance measures can be displayed in graphical or text form (on the screen or saved to file) for subsequent analysis.

Disadvantages

This simulator consists of too many parameters to be considered during setting up the network topology. User must know well about these parameters before can use it. Besides, user needs to have a strong programming language, especially in C language in order to customize the simulator's components. This simulator onli can run on UNIX or LINUX platform that give platform limitation problem.

	Object Oriented	GUI	Multithread	Platform Independent
Omnet	✓	✓✓✓	X	X
NS2	✓	✓✓	✓	X
Real NS	X	✓	X	X
Insane	✓	✓	✓	X
NetSim++	X	✓	X	X

Opnet++	✓	✓✓	X	X
NIST ATM/ HFC	X	✓✓	X	X

Table 2. 6 Comparison of network simulation

2.5 Summary

This chapter has covered the review of different types of network technologies. The review includes the details information about TCP/IP, DiffServ network and MPLS network.

The next chapter will discuss more details in traffic conditioner for DiffServ Network. The main purpose of this chapter is to discuss the traffic conditioner of Three Colour Marker.

CHAPTER 3 TRAFFIC CONDITIONER – TCM

A DS-Domain is made up of DS Boundary nodes (Ingress & Egress) and DS Interior nodes (in the core). Boundary nodes act as a demarcation point between the DS-Domain and the non-DS-aware (L2-LAN, etc.) network. Typically, the DS Boundary node performs traffic conditioning. Figure 3.1 illustrates the typical traffic conditioner at the edge of a DS-Domain. A DS Internal node enforces the appropriate PHB by employing policing or shaping techniques, and sometimes re-marking out of profile packets, depending on the policy or the SLA.

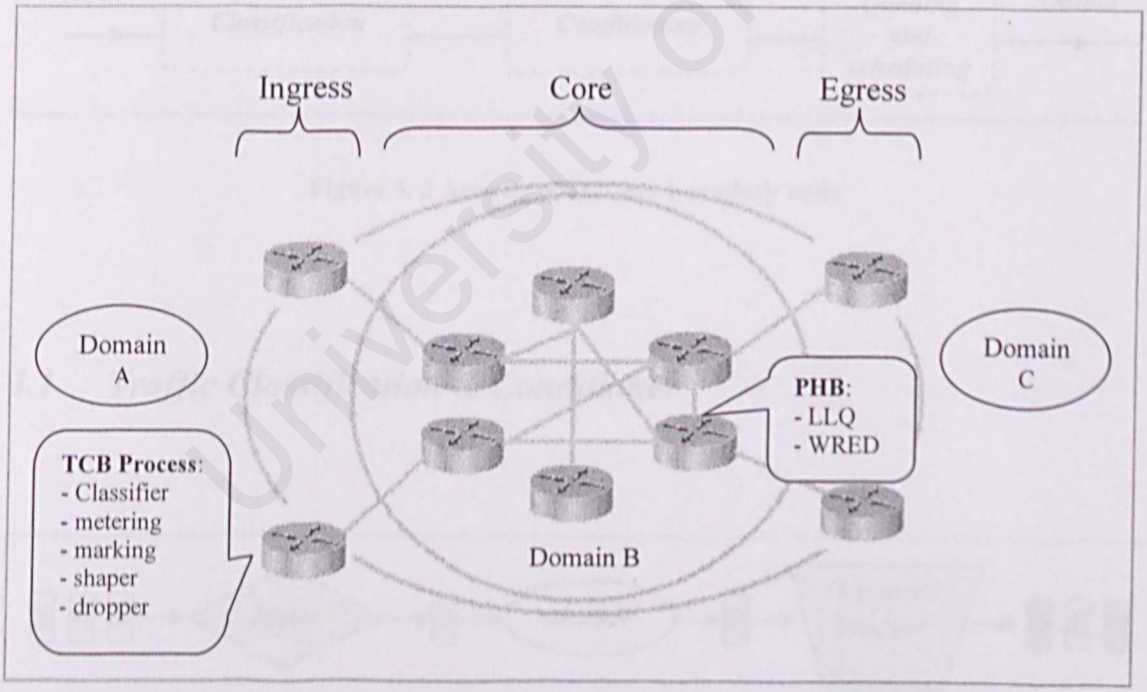


Figure 3. 1 DiffServ Network

The traffic profile for a customer is defined in her SLA with her ISP and it specifies the type and amount of traffic for which the ISP is willing to provide QoS guarantees. The traffic profile is defined in terms of the average rate, peak rate, burst size etc. Usually in-profile and out-of-profile packets are conditioned differently at the boundary nodes. The two types of packets might be marked differently, different accounting and billing procedures could be used for the two types of packets or the out-of-profile packets might be delayed or dropped to ensure that the outgoing traffic stream matches the traffic profile better. The traffic classifier and conditioner carry out all these activities. A logical diagram of which is shown below.

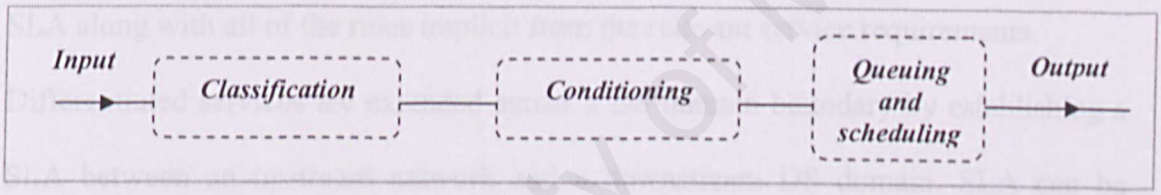


Figure 3. 2 Activity of Diffserv boundary node

3.1 *Traffic Classification & Conditioner*

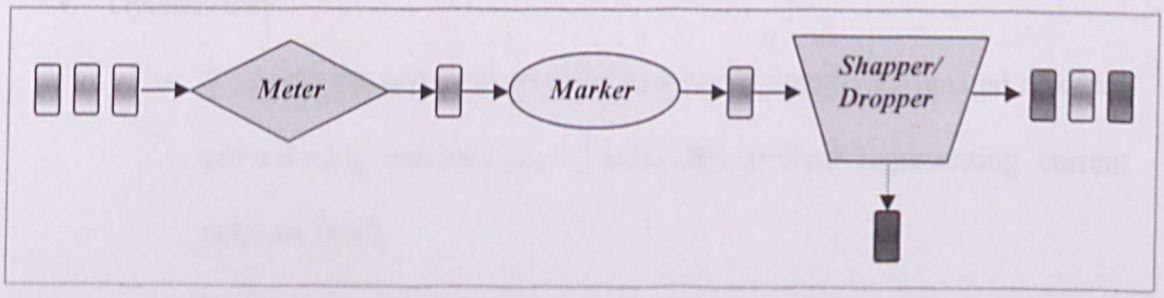


Figure 3. 3 Traffic conditioner functionality

A traffic conditioner is referred to as a set of components that may include a meter, marker, shaper, and a dropper. Traffic conditioners are usually located within boundary nodes (ingress, egress), but may also be located in nodes within the interior of a DS domain. The traffic conditioner bases its actions on the traffic profile that has been contracted between the user and the provider. An agreement specifying classifier rules and any corresponding traffic profiles and metering, marking, discarding and/or shaping rules which are to apply to the traffic streams selected by the classifier. A TCA encompasses all of the traffic conditioning rules explicitly specified within a SLA along with all of the rules implicit from the relevant service requirements.

Differentiated services are extended across a DS domain boundary by establishing a SLA between an upstream network and a downstream DS domain. SLA can be implemented:

- Statically:
    - are fixed agreements which may identify periods of time during which a certain service level can be offered
- Dynamically:
    - Requires the provider network to adopt dynamic, automated resource provisioning mechanisms. (bandwidth brokers representing current network load)
    - Requires customer equipment to adapt to dynamic SLA using some kind of signaling. (*Dr. Ljiljana Trajkovic. 2003*)

Classifier

Classifier classifies packets according to source/ destination IP addresses, port and destination numbers, DSCP, etc. There are two types of classifier:

i. Behavior aggregate (BA) classifier

- Selects packets based solely on the DS Codepoint (DSCP) value
- Used in the interior nodes

ii. Multifield (MF) classifier

- Uses a combination of the following fields of packet header:
  - Source IP address
  - Destination IP address
  - Source port
  - Destination port
  - Protocol ID

Marker

Mark/remark (for out-of-profile packets that are not dropped immediately) the DS Codepoint (DSCP) in a packet based on well-defined rules. Marking may occur at different locations by user applications, first-hop router, and boundary router of a DS domain.

The traffic conditioner forms a key part of a differentiated services network. Its purpose is to apply conditioning functions on previously classified packets according

to a predefined profile, i.e. a traffic-conditioning specification (TCS). A traffic conditioner consists of one or more components, as follows:

Meter

Checks compliance to traffic parameters (e.g., Token Bucket) and passes results to marker and shaper/dropper to trigger action for in/out-of-profile packets. In-profile packets are allowed to enter the network while out-of-profile packets are further conditioned.

Shaper

Shaper shapes the traffic stream to bring it into compliance with a traffic profile. (Difference between the shaper and the marker is that the shaper really prevents the packet from entering the network until the stream conforms to the profile)

Dropper

Discards packets based on specified rules (e.g. when the traffic stream does not conform to its TCS).

Leaky Bucket

Leaky bucket was proposed by Turner in 1986 to regulate the traffic from a source. It can be used as a policing device with a counter and it can be upgraded to a shaper with the addition of a buffer. When used as a shaper, a source in the network may

contain an interface with a leaky bucket, i.e. a finite internal queue. All the packets to be transmitted by the host are included in the queue. At a given time, the queue may be empty, partially filled or full. When the queue becomes full, the newly arriving packets are discarded. The buffer may drain onto the subnet either by some number of packets per unit time or by some number of bytes per unit time (helpful if packets vary greatly in size). A buffer converts an unregulated, bursty traffic flow into a regulated, smooth, predictable flow. The buffer is inserted between a traffic source and the subnet and it acts like a single server queue with a finite queue length.

Token Bucket

The leaky bucket shaper enforces a rigid output pattern at the average rate, no matter how bursty the traffic is. It eliminates the burstiness in the input and produces a smooth output. For many applications, it is better to allow output to speed up somewhat when a large burst arrives, so a variant of leaky bucket was developed. This variant is the token bucket. Instead of having fixed output rate regardless of variations in the traffic rate, the bucket is filled with tokens at a certain rate. A packet must grab and destroy a token to leave the bucket. If there are not enough tokens available, packets are not discarded rather they wait for an available token. If a burst arrives, it is allowed to pass through if enough tokens have accumulated in TB. Therefore the amount of burst allowed would be proportional to the elapsed time before the burst. This scheme is a compromise between controlling congestion and fulfilling source needs.

3.2 *Usage Parameters*

There are five traffic parameter have to be configure in the three colour marker.

3.2.1 CIR

Committed information rate is the contracted average sending rate. In srTCM and trTCM, it is used as green token arriving rate.

3.2.2 PIR

Peak information rate is the maximum contracted sending rate. In trTCM marking scheme, it is used as the yellow token bucket size.

3.2.3 CBS

Committed Burst Size, is the contracted traffic burst size. In srTCM and trTCM, it is used as the green token bucket size.

3.2.4 PBS

Peak Burst Size, is the yellow token bucket size in the trTCM scheme.

3.2.5 EBS

Excess Burst Size, is the excess token bucket (used to hold excess tokens) size in the srTCM scheme.

3.3 Single Rate Three Colour Marker (srTCM)

3.3.1 Overview

As described in, the Single-Rate Three-Colour Marker (srTCM) meters an IP packet stream and marks its packets green, yellow, or red before admitting them in a diffserv domain. Marking is based on a Committed Information Rate (CIR) and two associated burst sizes, a Committed Burst Size (CBS) and an Excess Burst Size (EBS). A packet is marked green if it doesn't exceed the CBS, yellow if it does exceed the CBS, but not the EBS, and red otherwise.

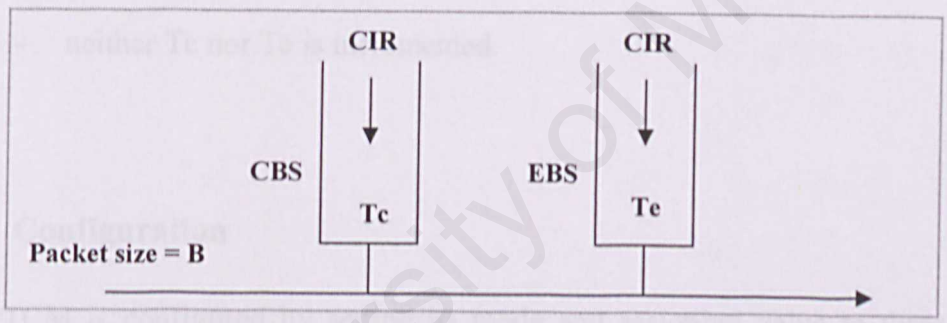


Figure 3. 4 srTCM

The srTCM is useful, for example, for ingress policing of a service. In such a policing scheme, only the length, not the peak rate, of the burst determines service eligibility. The meter can operate in colour blind mode in which it assumes that the packet stream is uncoloured, or in colour aware mode in which the meter assumes that the packets have already been coloured by some previous entity. The colour is coded in the DS field of the packet in the PHB specified manner. The behavior of the meter is specified in terms of its mode and two token buckets (C & E), which both share the common rate CIR. After the meter the marker should reflects the metering result by

3.3 *Single Rate Three Colour Marker (srTCM)*

3.3.1 Overview

As described in, the Single-Rate Three-Colour Marker (srTCM) meters an IP packet stream and marks its packets green, yellow, or red before admitting them in a diffserv domain. Marking is based on a Committed Information Rate (CIR) and two associated burst sizes, a Committed Burst Size (CBS) and an Excess Burst Size (EBS). A packet is marked green if it doesn't exceed the CBS, yellow if it does exceed the CBS, but not the EBS, and red otherwise.

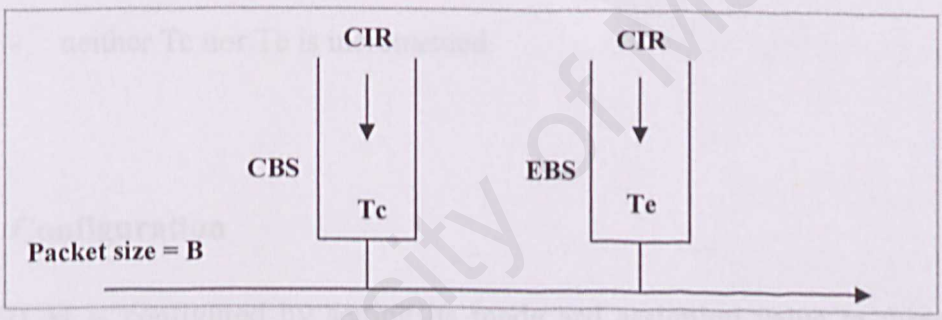


Figure 3. 4 srTCM

The srTCM is useful, for example, for ingress policing of a service. In such a policing scheme, only the length, not the peak rate, of the burst determines service eligibility. The meter can operate in colour blind mode in which it assumes that the packet stream is uncoloured, or in colour aware mode in which the meter assumes that the packets have already been coloured by some previous entity. The colour is coded in the DS field of the packet in the PHB specified manner. The behavior of the meter is specified in terms of its mode and two token buckets (C & E), which both share the common rate CIR. After the meter the marker should reflects the metering result by

setting the DS field of the packet to a particular codepoint. In case of AF PHB, the colour can be coded as the drop precedence of the packet. The maximum size of the token bucket C is CBS and the maximum size of the token bucket E is EBS. (*Dr. Ljiljana Trajkovic. 2003)*

The token buckets C and E are initially (at time 0) full, i.e., the token count Tc(0)=CBS and the token count Te(0)=EBS. Thereafter, the token counts Tc and Te are updated CIR times per second as follows:

- If Tc is less than CBS, Tc is incremented by one, else
- if Te is less then EBS, Te is incremented by one, else
- neither Tc nor Te is incremented.

3.3.2 Configuration

The srTCM is configured by setting its mode and assigning value to three traffic parameter as describe in chapter 3.3, which includes:

- Committed Information Rate(CIR)
- Committed Burst Size (CBS)
- Excess Burst Size (EBS)

3.3.3 Metering

The behaviour of the meter is specified in terms of its mode and two token bucket (token bucket 'C' and token bucket 'E'). C and E are sharing the common rate CIR.

CBS is the maximum size of token count C (Tc), while EBS is the maximum size of token count E (Te).

At the beginning, the token bucket C and E are initially full (Tc(0)=CBS; Te(0)=EBS). Thereafter, Tc and Te will be updated with rate CIR times per second as below:

- If Tc<CBS, Tc is incremented by one, else
- If Te is less than EBS, Te is incremented by one, else
- Neither Tc nor Te is incremented

The metering behaviour will be described clearer in each of the operation mode.

3.3.4 srTCM Operation Mode

srTCM can be configured to operate in two mode:

- Colour blind mode: the packet is assumed uncoloured
- Colour aware mode: the packet is assumed coloured by entity before

Colour Blind Mode of srTCM

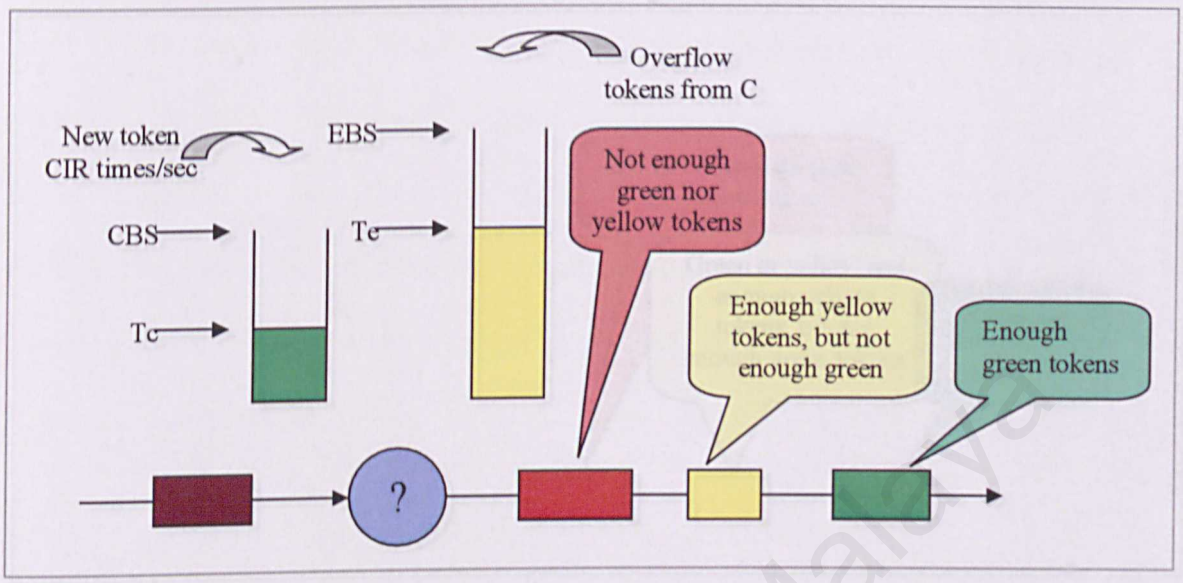


Figure 3. 5 Colour blind mode of srTCM

If the srTCM is configured to operate in the Colour-Blind mode; when a packet of size B bytes arrives at time t, the following happens:

- If Tc(t)-B >= 0, the packet is green and Tc is decremented by B down to the minimum value of 0, else
- if Te(t)-B >= 0, the packets is yellow and Te is decremented by B down to the minimum value of 0, else
- the packet is red and neither Tc nor Te is decremented.

Colour Aware Mode of srTCM

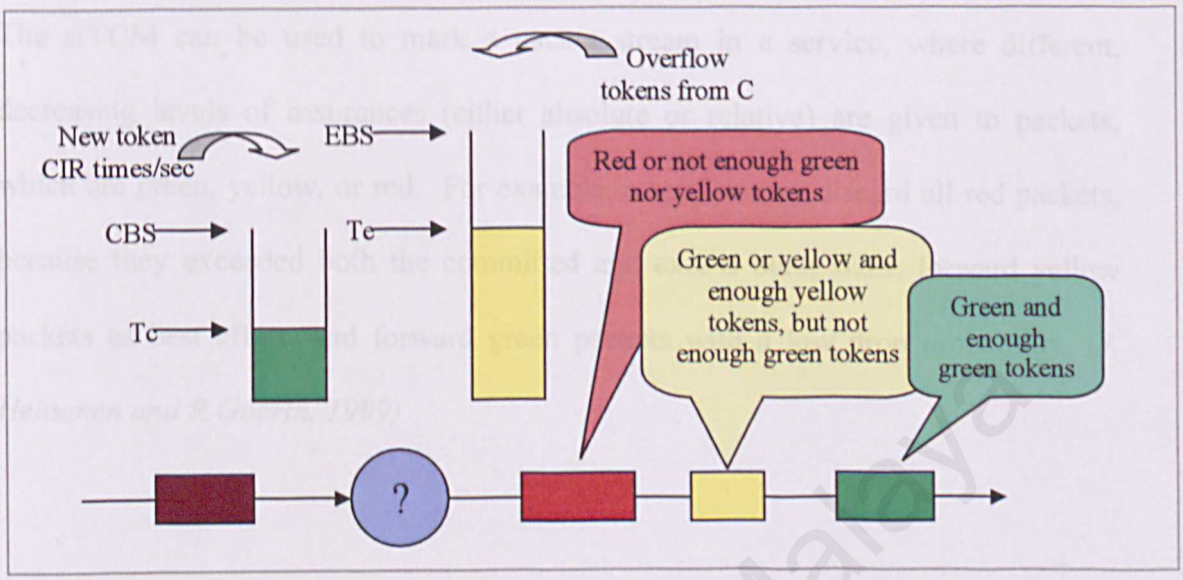


Figure 3. 6 Colour aware mode of srTCM

if the srTCM is configured to operate in the Colour-Aware mode; when a packet of size B bytes arrives at time t, the following happens:

- If the packet has been precoloured as green and Tc(t)-B >= 0, the packet is green and Tc is decremented by B down to the minimum value of 0, else
- If the packet has been precoloured as green or yellow and if
- Te(t)-B >= 0, the packets is yellow and Te is decremented by B down to the minimum value of 0, else
- the packet is red and neither Tc nor Te is decremented.

3.3.5 Service Example

The srTCM can be used to mark a packet stream in a service, where different, decreasing levels of assurances (either absolute or relative) are given to packets, which are green, yellow, or red. For example, a service may discard all red packets, because they exceeded both the committed and excess burst sizes, forward yellow packets as best effort, and forward green packets with a low drop probability. (*J. Heinanen and R.Guerin. 1999)*

3.4 Two Rate Three Colour Marker (trTCM)

3.4.1 Overview

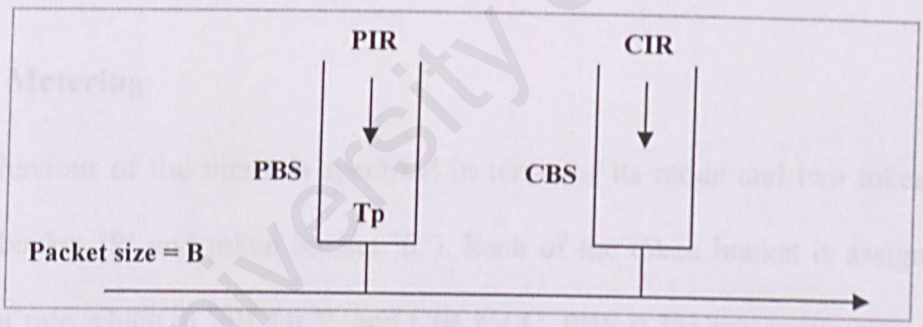


Figure 3. 7 trTCM

trTCM is a marker applicable to the three drop precedence, and is proposed as a component of the traffic conditioner. The trTCM meters an IP packet stream and marks its packets based on two rates, the Peak Information Rate (PIR) and the Committed Information Rate (CIR), and on their associated burst sizes, the Peak Burst Size (PBS) and the Committed Burst Size (CBS) respectively. The two meters

of the trTCM use token buckets with parameters (CIR, CBS) and (PIR, PBS). (*Dr. Ljiljana Trajkovic. 2003)*

3.4.2 Configuration

The trTCM is configured by setting its mode and assigning value to four traffic parameter as describe in chapter 3.3, which includes:

- Peak Information Rate (PIR)
- Committed Information Rate(CIR)
- Peak Burst Size (PBS)
- Committed Burst Size (CBS)

3.4.3 Metering

The behaviour of the meter is specified in terms of its mode and two token bucket (token bucket 'P' and token bucket 'C'). Each of the token bucket is assigned with different rate which is PIR for P, and CIR for C. PBS is the maximum size of token count P (Tp), while CBS is the maximum size of token count C (Tc).

At the beginning, the token bucket P and C are initially full (Tp(0)=PBS; Tc(0)=CBS). Thereafter, Tp is incremented by one PIR times per second up to PBS and Tc is incremented by one CIR times per second up to CBS.

The metering behaviour will be described clearer in each of the operation mode.

3.4.4 trTCM Operation Mode

Like srTCM, trTCM also can be configured to operate in two mode:

- Colour blind mode: the packet is assumed uncoloured
- Colour aware mode: the packet is assumed coloured by entity before

Colour Blind Mode of trTCM

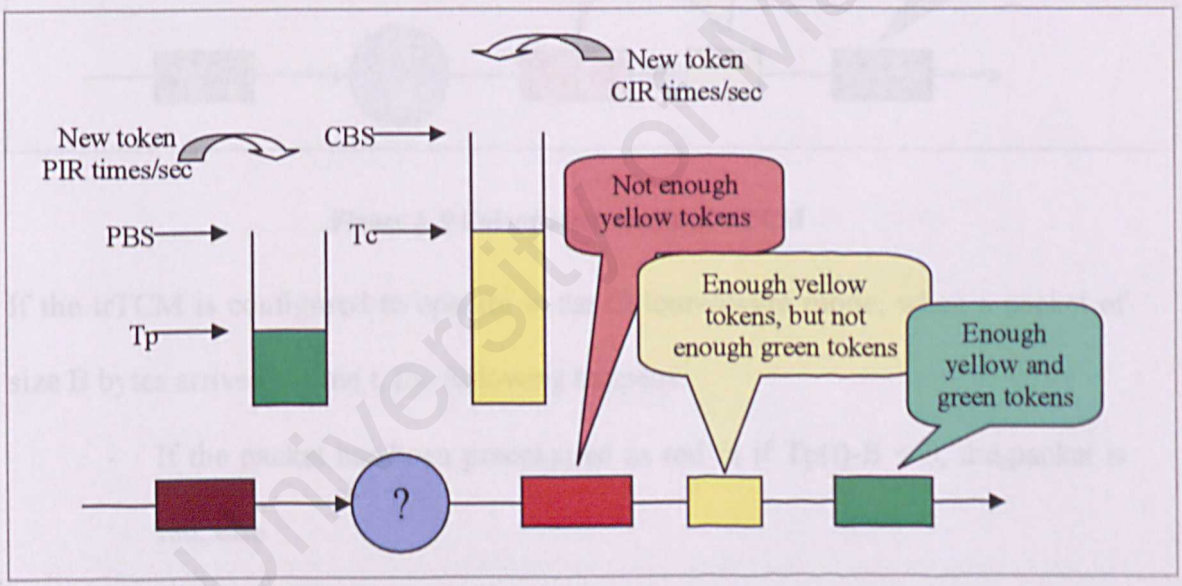


Figure 3. 8 Colour blind mode of trTCM

If the trTCM is configured to operate in the Colour-Blind mode; when a packet of size B bytes arrives at time t, the following happens:

- If $Tp(t)-B < 0$, the packet is red, else
- if $Tc(t)-B < 0$, the packet is yellow and Tp is decremented by B, else

- the packet is green and both Tp and Tc are decremented by B.

Colour Aware Mode of trTCM

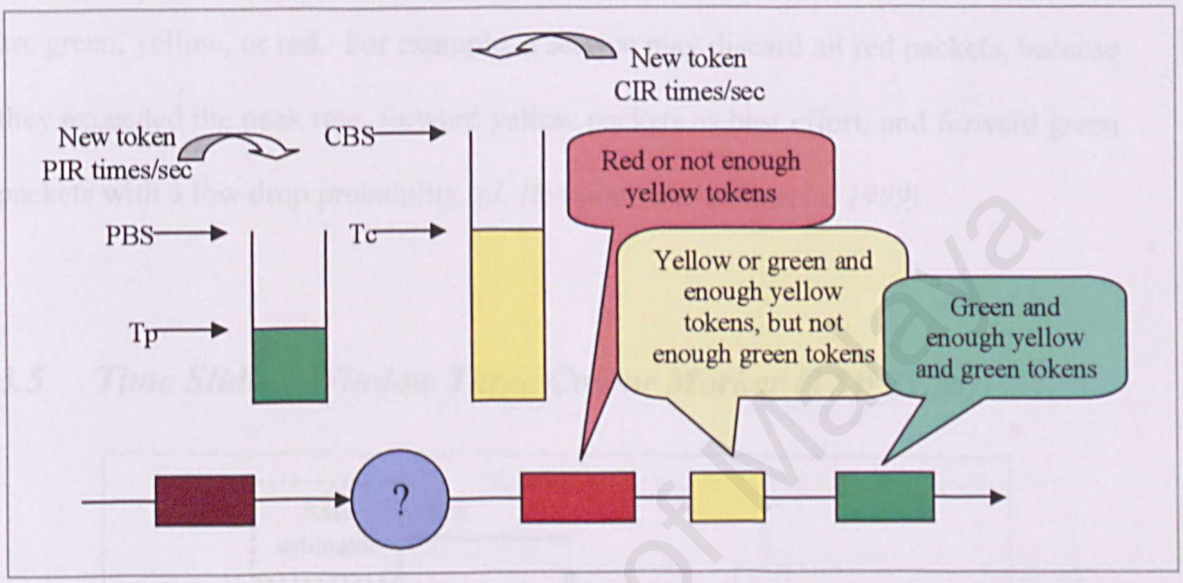


Figure 3. 9 Colour aware mode of trTCM

If the trTCM is configured to operate in the Colour-Aware mode; when a packet of size B bytes arrives at time t, the following happens:

- If the packet has been precoloured as red or if Tp(t)-B < 0, the packet is red, else
- if the packet has been precoloured as yellow or if Tc(t)-B < 0,the packet is yellow and Tp is decremented by B, else
- the packet is green and both Tp and Tc are decremented by B.

3.4.5 Service Example

The trTCM can be used to mark a IP packet stream in a service, where different, decreasing levels of assurances (either absolute or relative) are given to packets which are green, yellow, or red. For example, a service may discard all red packets, because they exceeded the peak rate, forward yellow packets as best effort, and forward green packets with a low drop probability. *(J. Heinanen and R.Guerin. 1999)*

3.5 Time Sliding Window Three Colour Marker (TSWTCM)

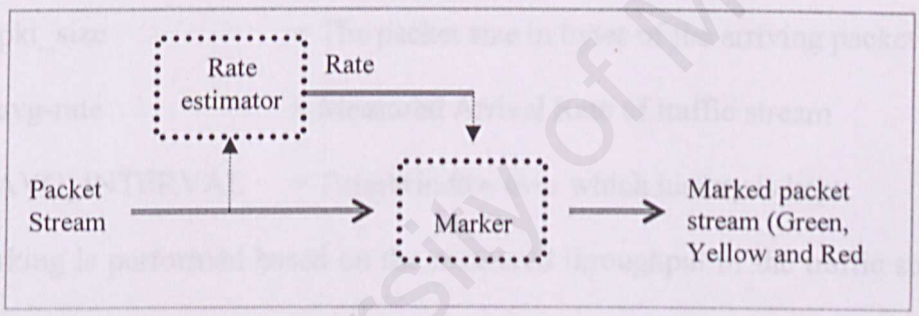


Figure 3. 10 Block diagram for tswTCM

The TSWTCM consists of two independent components: a rate estimator, and a marker to associate a colour (drop precedence) with each packet. The marker mark packets of IP traffic stream with colour of red, yellow or green uses the algorithm as follows:

AVG_INTERVAL = a constant;

avg-rate = CTR;

t-front = 0;

Upon each packet's arrival, the rate estimator updates its variables:

Bytes_in_win = avg-rate * AVG_INTERVAL;

New_bytes = Bytes_in_win + pkt_size;

avg-rate = New_bytes/(now - t-front + AVG_INTERVAL);

t-front = now;

Where:

now = The time of the current packet arrival

pkt_size = The packet size in bytes of the arriving packet

avg-rate = Measured Arrival Rate of traffic stream

AVG_INTERVAL = Time window over which history is kept

The marking is performed based on the measured throughput of the traffic stream as compared against the Committed Target Rate (CTR) and the Peak Target Rate (PTR).

- If the packet is contributing to sending rate below or equal to the CTR, the packet is green, else
- If the packet is contributing to the portion of the rate between the CTR and PTR, the packet is yellow, else
- If the packet is causing the rate to exceed PTR, mark it with red.

The rate estimator provides an estimate of the running average bandwidth. It takes into account burstiness and smoothes out its estimate to approximate the longer-term measured sending rate of the traffic stream.

The marker uses the estimated rate to probabilistically associate packets with one of the three colours. Using a probabilistic function in the marker is beneficial to TCP flows as it reduces the likelihood of dropping multiple packets within a TCP window. The marker also works correctly with UDP traffic, i.e., it associates the appropriate portion of the UDP packets with yellow or red colour marking if such flows transmit at a sustained level above the contracted rate.

The colour of the packet is translated into a DS field packet marking. The colours red, yellow and green translate into DS codepoints representing drop precedence 2, 1 and 0 of a single AF class respectively.

3.6 Experiment on Three type of Three Colour Marker

This is an experiment result using NS-2 simulator with Single Rate Three-Colour Marker (srTCM), Two Rate Three-Colour Marker (trTCM), and Time Sliding Window (TSW) marking scheme. This experiment is a diffserv based mobile nodes experiment.

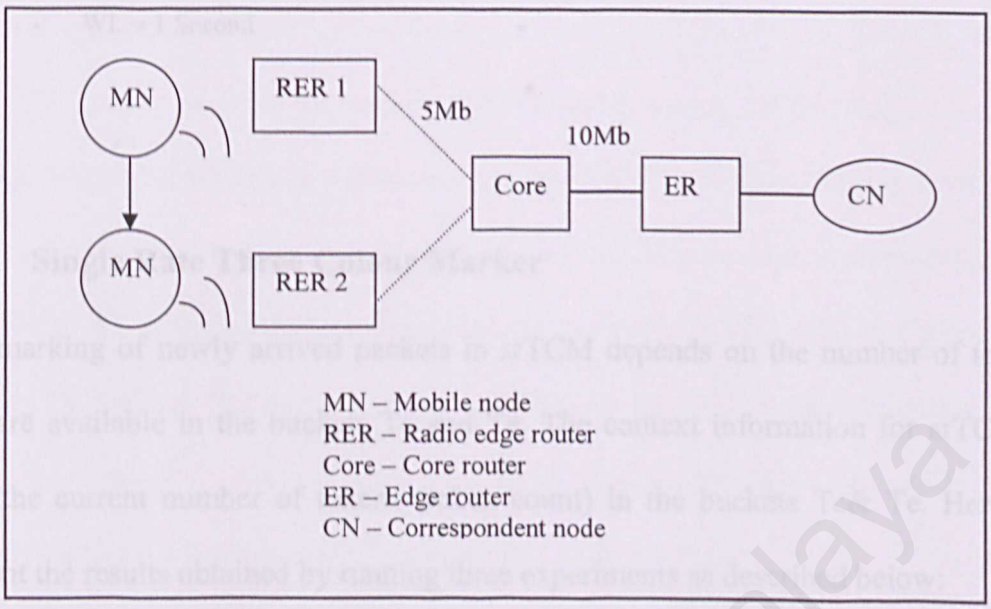


Figure 3. 11 Experimental setup

Parameters for srTCM:

- CIR = 1Mbps
- CBS = 10000 bytes
- EBS = 30000 bytes

Parameters for srTCM:

- CIR = 1Mbps
- CBS = 10000 bytes
- PIR = 2Mbps
- PBS = 30000 bytes

Parameters for tswTCM:

- CTR = 1Mbps
- PTR = 2Mbps

- WL = 1 Second

3.6.1 Single Rate Three Colour Marker

The marking of newly arrived packets in srTCM depends on the number of tokens that are available in the buckets Tc and Te. The context information for srTCM is thus the current number of tokens (token count) in the buckets Tc& Te. Here we present the results obtained by running three experiments as described below:

1. In the first experiment packet statistics are collected for three seconds at RER1 without moving the MN to RER2, hence this case is without handoff.
2. In the second experiment, packet statistics are collected for three seconds at RER2 without handing over the token count computed at RER1 to RER2; hence this case shows the data without context transfer. The handoff takes place at 40th second and the handoff latency is assumed to be zero.
3. In the third experiment, packet statistics are collected for three seconds at RER2 with handing over the token count computed at RER1 to RER2 at 40^{th} second (handoff time), hence this case shows the data with context transfer. The handoff takes place at 40th second and the handoff latency is assumed to be zero, context is transferred in zero time.

Analysis of srTCM Results

Figures 3.12 to 3.14 show the packet distribution for green, yellow, and red packets that are plotted by collecting statistics at the interval of 0.5 second within a period of three seconds from the 40th to 43rd seconds. In these figures, abscissa represents time from 40 to 43 seconds, while ordinate shows the packet distribution for green (G), yellow (Y), and red (R) packets. Figure 3.12 shows the packet distribution collected during experiment 4 (without handoff). The marking is almost stable, and it shows stability right from 40th second (starting point of the graph).

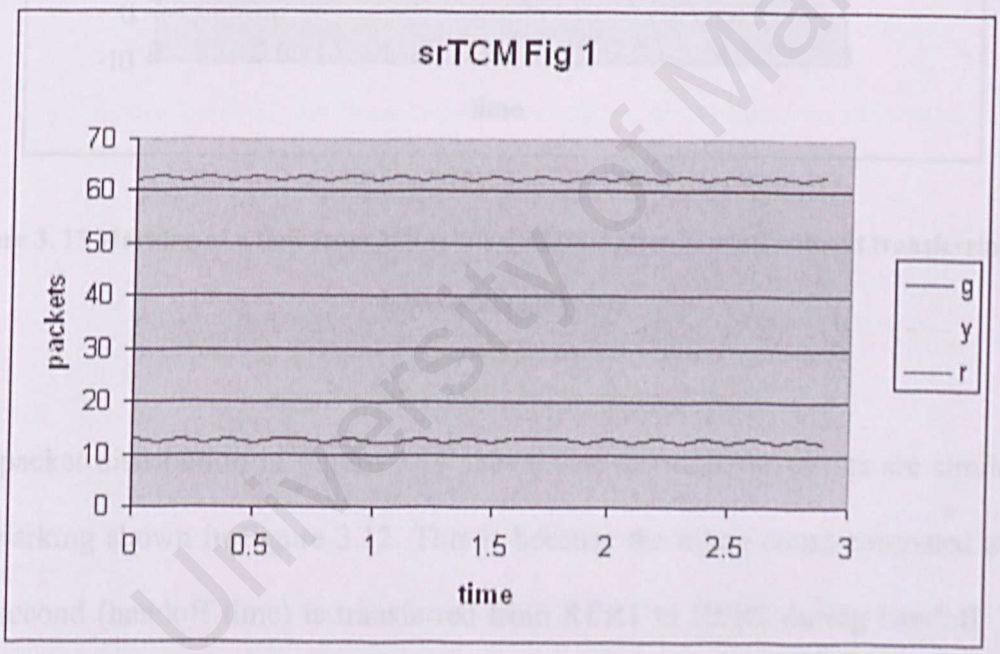


Figure 3. 12 Marking of a flow from MN to CN at RER 1 without handoff using srTCM

In Figure 3.13 the handoff takes place at t = 40th second. In this case the token count of Tc and Te is not transferred from RER1 to RER2 during handoff. As a result, the srTCM starts marking the incoming traffic from the initial window setting. Tc & Te are set to the maximum value CBS and EBS. This allows a burst equal to CBS to pass

as green and EBS as yellow at the beginning of the handoff as to shown in Figure 3.13.

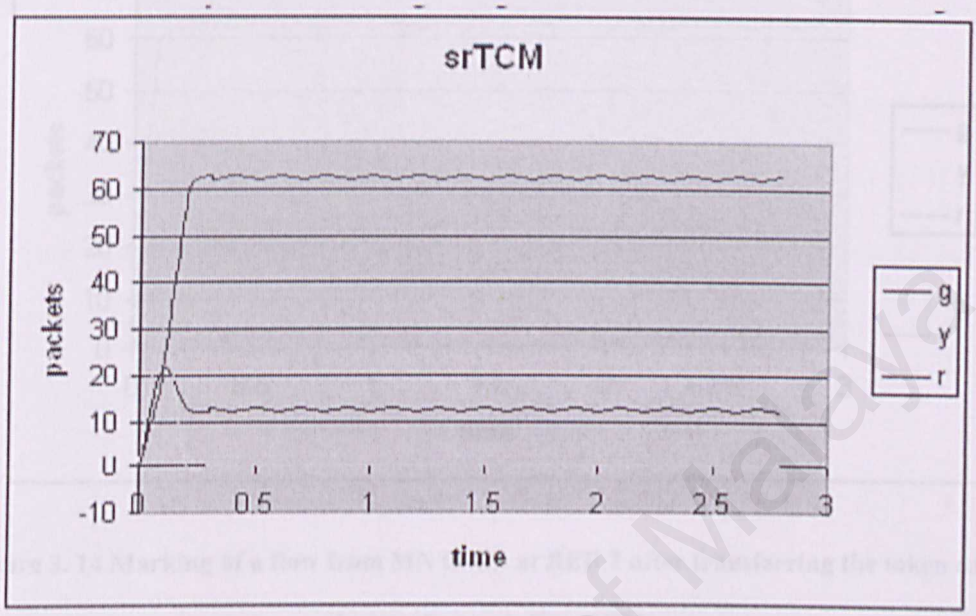


Figure 3. 13 Marking of a flow from MN to CN at RER 2 after handoff without transferring the token count (srTCM)

The packet distribution in Figure 3.14 shows that the marking results are similar to the marking shown in Figure 3.12. This is because the token count computed at the 40th second (handoff time) is transferred from RER1 to RER2 during handoff. This calibrates the mater at RER2 to get results closer to the results in experiment 4.

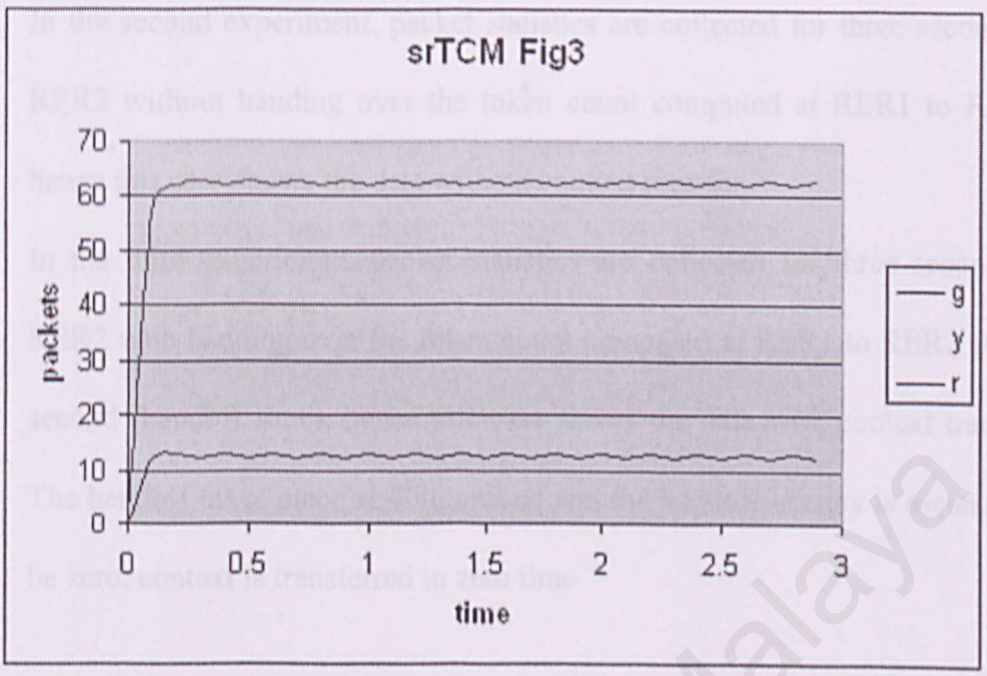


Figure 3. 14 Marking of a flow from MN to CN at RER 2 after transferring the token count during the handoff (srTCM)

3.6.2 Two Rate Three Colour Marker

The marking of a new packet in trTCM depends on the number of tokens that are available in the buckets Tc and Tp. The context information for trTCM is thus the token count in the buckets Tc& Tp. This section presents the results obtained by running three experiments as described below:

1. In the first experiment packet statistics are collected for three seconds at RER1 without moving the MN to RER2, hence this case shows the data without handoff.

2. In the second experiment, packet statistics are collected for three seconds at RER2 without handing over the token count computed at RER1 to RER2; hence this case shows the data without context transfer.
3. In the third experiment, packet statistics are collected for three seconds at RER2 with handing over the token count computed at RER1 to RER2 at 40th second (handoff time), hence this case shows the data with context transfer. The handoff takes place at 40th second and the handoff latency is assumed to be zero, context is transferred in zero time.

Analysis of trTCM Results

Figures 3.15 to 3.17 show the packet distribution for green, yellow, and red packets that are plotted by collecting statistics at the interval of 0.1 second within a period of three seconds from the 40th to 43rd seconds. In these figures abscissa represents time from 40th to 43rd second, while ordinate shows the packet distribution for green (G), yellow (Y), and red (R) packets. Figure 3.15 shows the packet distribution collected during experiment 7 (without handoff). The marking is almost stable, and it shows stability right from 40th second (starting point of the graph).

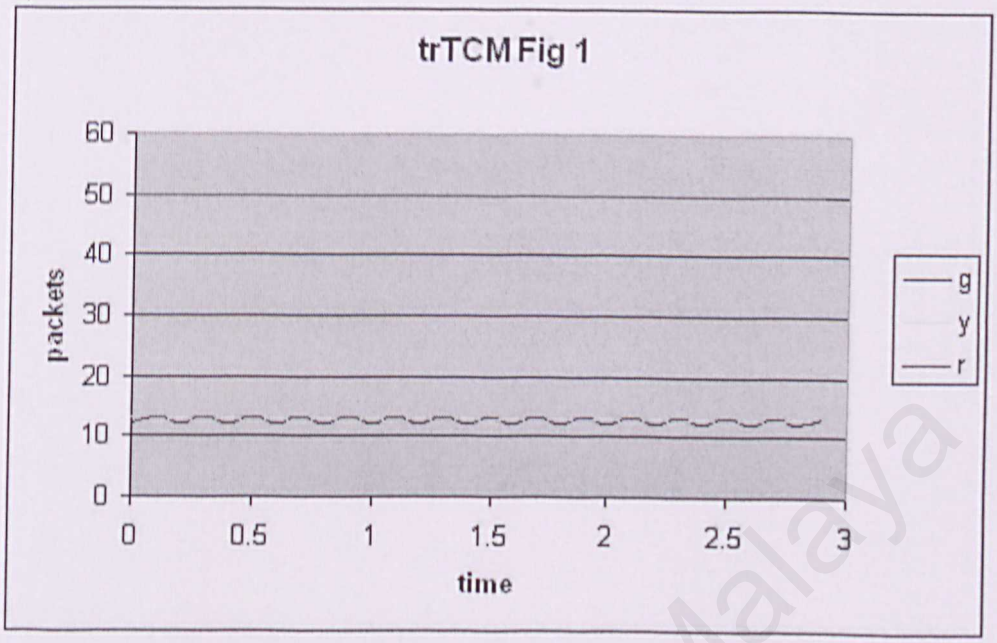


Figure 3. 15 Marking of a flow from MN to CN at RER 1 without handoff (trTCM)

In Figure 3.16 the handoff takes place at t = 40th second. In this case the token account is not transferred from RER1 to RER2 during handoff. As a result the trTCM starts marking the incoming traffic from the initial window setting. Tc & Tp set to the maximum value CBS &PBS this allows a burst equal to CBS to pass as green and PBS as yellow at the beginning of the handoff as to shown in Figure 3.16.

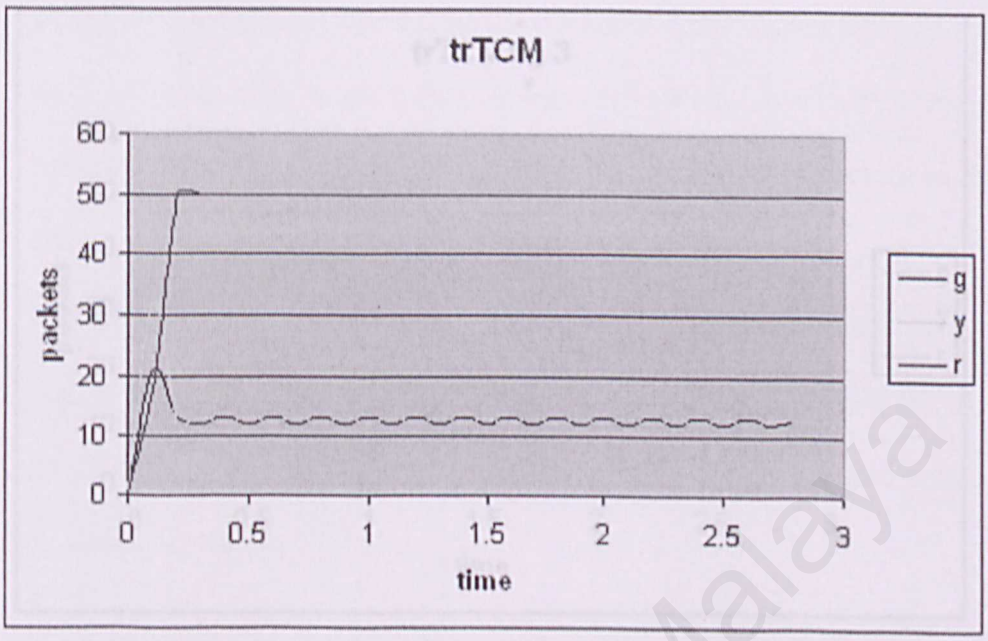


Figure 3. 16 Marking of a flow from MN to CN at RER 2 after handoff without transferring the token count (srTCM)

The packet distribution in Figure 3.17 shows that the marking results are similar to the marking shown in Figure 3.15. This is because the token count of Tc &Tp computed at the 40th second (handoff time) is transferred from RER1 to RER2 during handoff. This calibrates the mater at RER2 to get results closer to the results in experiment 7.

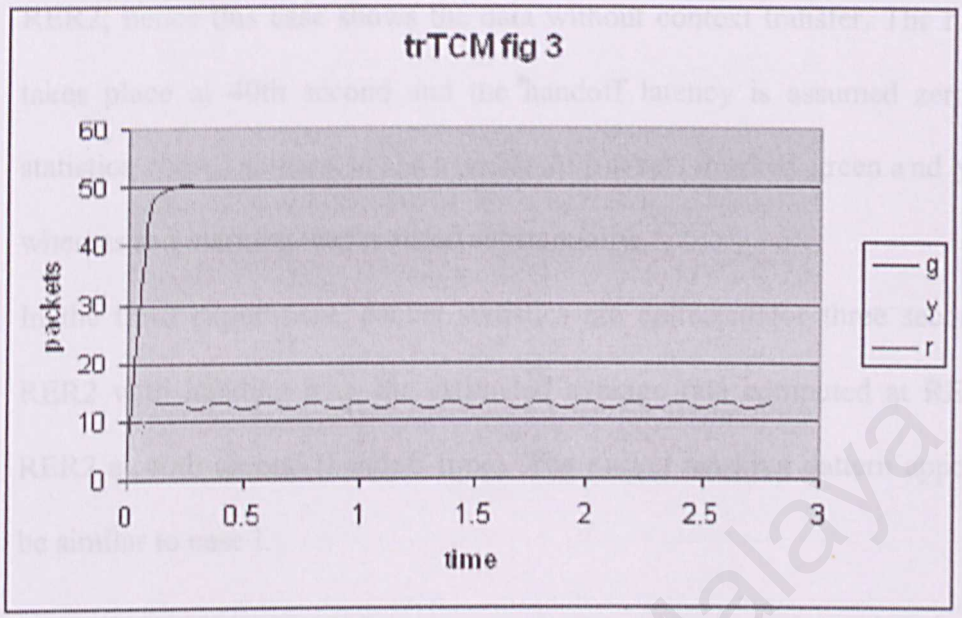


Figure 3. 17 Marking of a flow from MN to CN at RER 2 after transferring the token count during the handoff (trTCM)

3.6.3 Time Sliding Window TSW

A TSW meter estimates average packet arrival rate by including both the rate of newly arrived packet and the rate estimated within a window of history. The context information for TSW is thus the current average estimated rate. This section present the results obtained by running three experiments using TSW as described below:

1. In the first experiment packet statistics are collected for three seconds at RER1 without moving the MN to RER2, hence this is the case without handoff.
2. In the second experiment, packet statistics are collected for three seconds at RER2 without handing over the estimated average rate computed at RER1 to

RER2; hence this case shows the data without context transfer. The handoff takes place at 40th second and the handoff latency is assumed zero. The statistics show increase in the number of packets marked green and yellow whereas red marking was reduced substantially.

3. In the third experiment, packet statistics are collected for three seconds at RER2 with handing over the estimated average rate computed at RER1 to RER2 at 40th second (handoff time). The packet marking pattern appears to be similar to case I.

Analysis of TSW Results

Figures 3.18 to 3.20 show the packet distribution for green, yellow, and red packets that are plotted by collecting statistics at the interval of 0.5 seconds within a period of three seconds from the 40th to 43rd seconds. In these figures, abscissa is the time from 40th to 43rd seconds, while ordinate shows the packet distribution for green (G), yellow (Y), and red (R) packets. Figure 3.18 shows the packet distribution collected during experiment 1 (without handoff). The marking is almost stable, and it shows stability right from 40th second (starting point of the graph), because of the effect of prior estimated average rate.

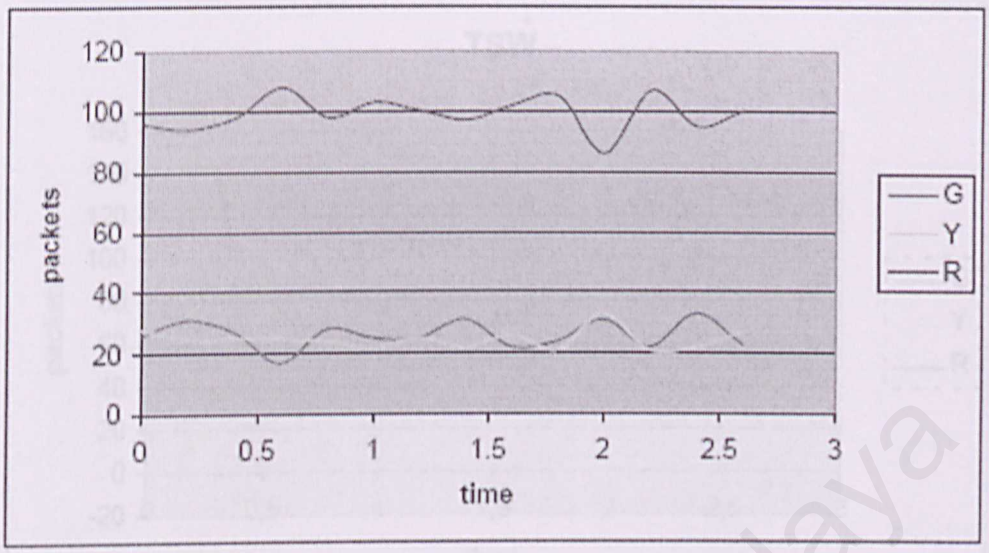


Figure 3. 18 Marking of a flow from MN to CN at RER 1 without handoff

In Figure 3.19, the handoff takes place at t = 40th second. In this case the estimated average rate is not transferred from RER1 to RER2 during handoff. As a result the TSW starts marking the incoming traffic from the initial window setting, that is using CTR as the past estimated average rate. The average rate is updated at the arrival of each packet therefore it takes a while for the average to reach a stable value that is reflective of the close approximation of the actual arrival rate. As long as the average rate is less than CTR all the packets are marked green, hence the number of the green packets is high at the beginning and then tapers off later (around one second later) to its stable value. Similar patterns are shown by the distribution of yellow and red packets as well. The graph shows that the rate estimate improves with time and consequently TSW marking reaches some sort of stability.

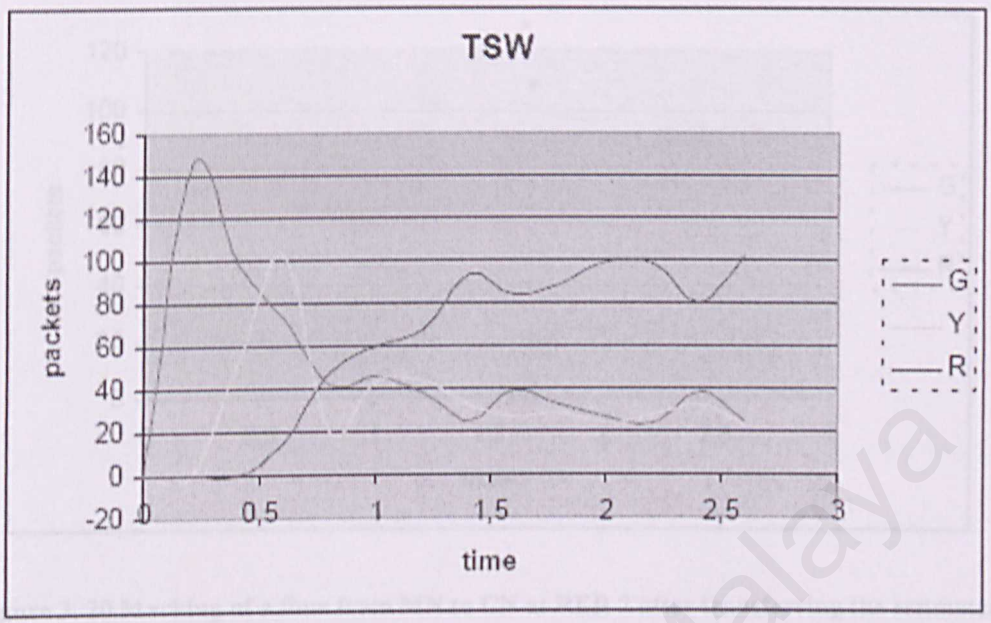


Figure 3. 19 Marking of a flow from MN to CN at RER 2 after handoff without transferring the estimated average

The packet distribution in Figure 3.20 shows that the marking reaches stability earlier than what it takes for the experiment 2 (as shown in Figure 6). This is because the estimated average rate computed at the 40th second (handoff time) is transferred from RER1 to RER2 during handoff thus calibrating the meter at RER2 to estimate the average rate closer to accuracy.

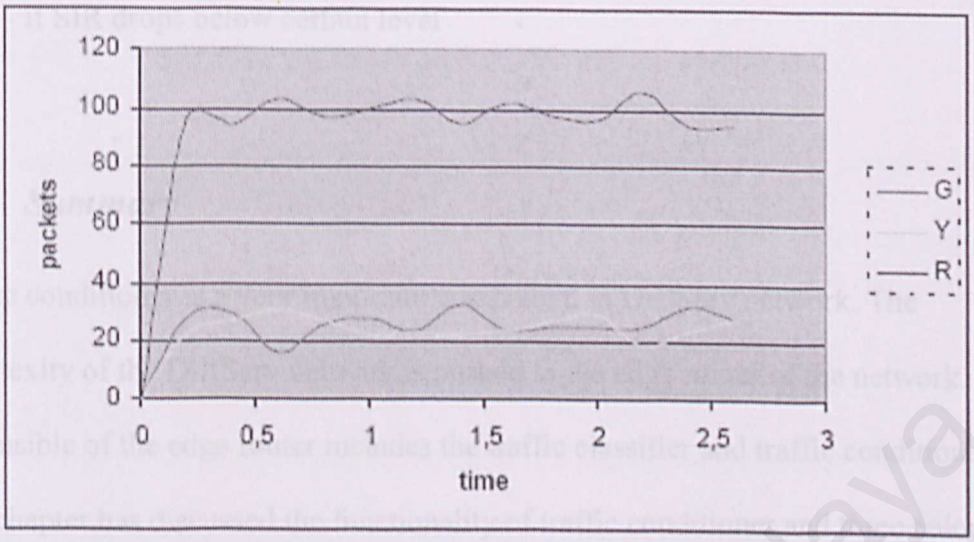


Figure 3. 20 Marking of a flow from MN to CN at RER 2 after transferring the estimated average during handoff

Handoff

In a wireless access network, an edge router called Radio Edge Router (RER) provides connectivity to mobile nodes. Any RER has certain geographic coverage area within which a mobile node can communicate to it. This coverage area is known as a cell. The procedure of maintaining a call in progress, while moving from one cell to another is called *handoff.* Handoff is caused by factors related to radio link and network management. Radio link related causes reflect the quality perceived by users. Some of the major variables affecting the service quality are received signal strength (RSS) and signal-to-interference ratio (SIR). Insufficient RSS and SIR reduce the service quality. Handoff is required in the following situations:

i. When the MN approaches the cell boundary (the RSS drops below a threshold)

ii. if SIR drops below certain level

3.7 Summary

Traffic conditioner is a very important component in DiffServ network. The complexity of the DiffServ network is pushed to the edge router of the network. The responsible of the edge router includes the traffic classifier and traffic conditioner. This chapter has discussed the functionality of traffic conditioner and three colour marker in details.

CHAPTER 4 SYSTEM ANALYSIS

This chapter provides an in depth analysis of UMJanetsim network simulator. The chapter begins with the overview of UMJanetsim network simulator's architecture and the UMJanetsim API.

Next, the analyses of the programming approach together with the hardware and software selection are included. The aim of system analysis is to collect the functional requirement and non-functional requirement of implementing the additional component in UMJanetsim network simulator.

4.1 UMJanetSim Architecture

The basic concepts used by JaNetSim are summarized as follows:

- discrete-event model,
- central simulation engine with a centralized event manager,
- simulation scenario consists of a finite number of interconnected components (simulation objects), each with a set of parameters (component properties),
- Simulation execution involves components sending messages among each other. A message is sent by scheduling an event (to happen some time later) for the target component.

With the above architecture, the simulator can simulate virtually "anything" that can be modeled by a network of components that send messages to one another. These concepts are adopted from the NIST ATM/HFC Network Simulator.

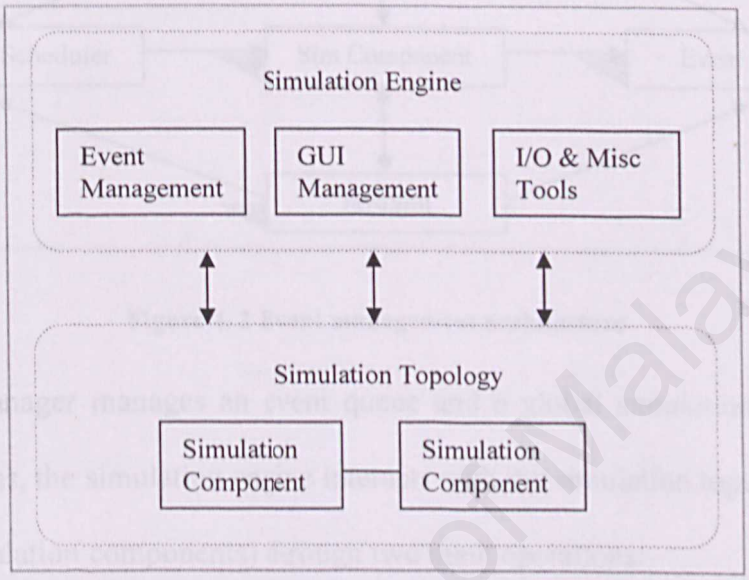


Figure 4. 1 Simulation engine & simulation topology

The simulation engine is the sole event manager and is responsible for the managing of all the user interface elements. The engine also provides convenient means for file saving and data logging, among other tools. The programmer is responsible for the development of simulation components that directly represent the intended system to be simulated.

Event Management Architecture

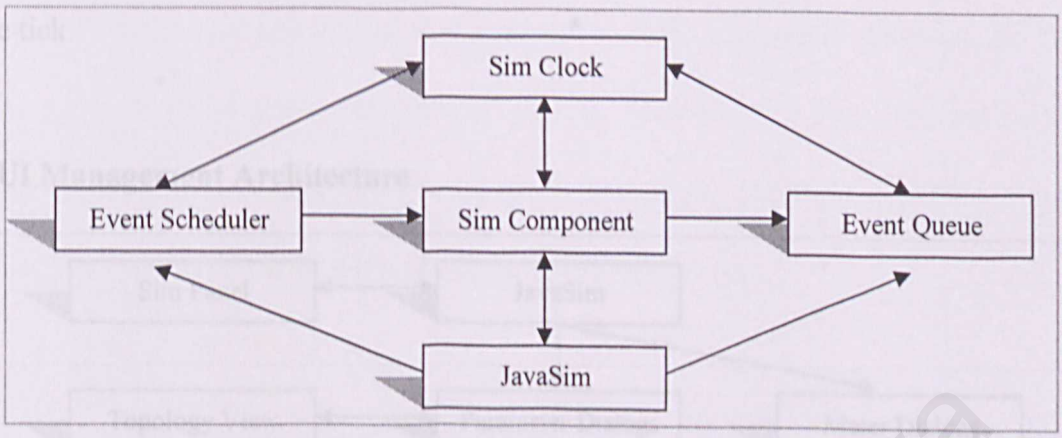


Figure 4. 2 Event management architecture

The event manager manages an event queue and a global simulation clock. When simulation runs, the simulation engine interacts with the simulation topology (consists of all the simulation components) through two main operations:

- A simulation component schedules an event for a target component (can be the source component itself) to be happen at a specific time
- The simulation engine invokes the event handler of the target component when that specific time is reached. The target component will react to the event according to its behavior.

The Simulation Time

Any event can happen at any time, up to the precision allowed by the granularity of the simulation clock. The simulation time is based on "ticks". The duration of a tick is configurable in the simulator. By default, a tick is equivalent to 10 nanoseconds. The

SimClock object provides helper methods for the conversion between real time and the tick.

GUI Management Architecture

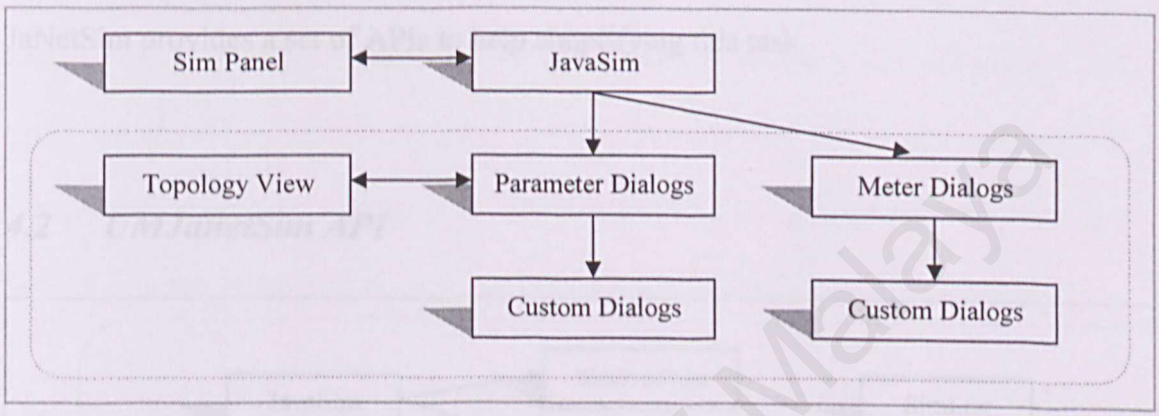


Figure 4. 3 GUI management architecture

It is possible to run the simulator in non-GUI mode once a scenario is built. The building of simulation scenarios is done using the GUI mode. The simulator has a main window, and within the main frame, there is a main simulation view area, which displays the simulation components and the connections among them. Basically the GUI manager handles all these automatically with only minimal effort from component developers.

A scenario is built by creating simulation components, connecting components and setting component parameters. The GUI manager handles three tasks.It will signal the relevant components when these operations happen so that a component can react accordingly.

Basically, building a component that needs no custom component graphics and only uses pre-built parameter types (integer, double, boolean etc.) does not require any GUI-related programming. For better GUI and more complex parameter types, JaNetSim provides a set of APIs to help simplifying this task.

4.2 *UMJanetSim API*

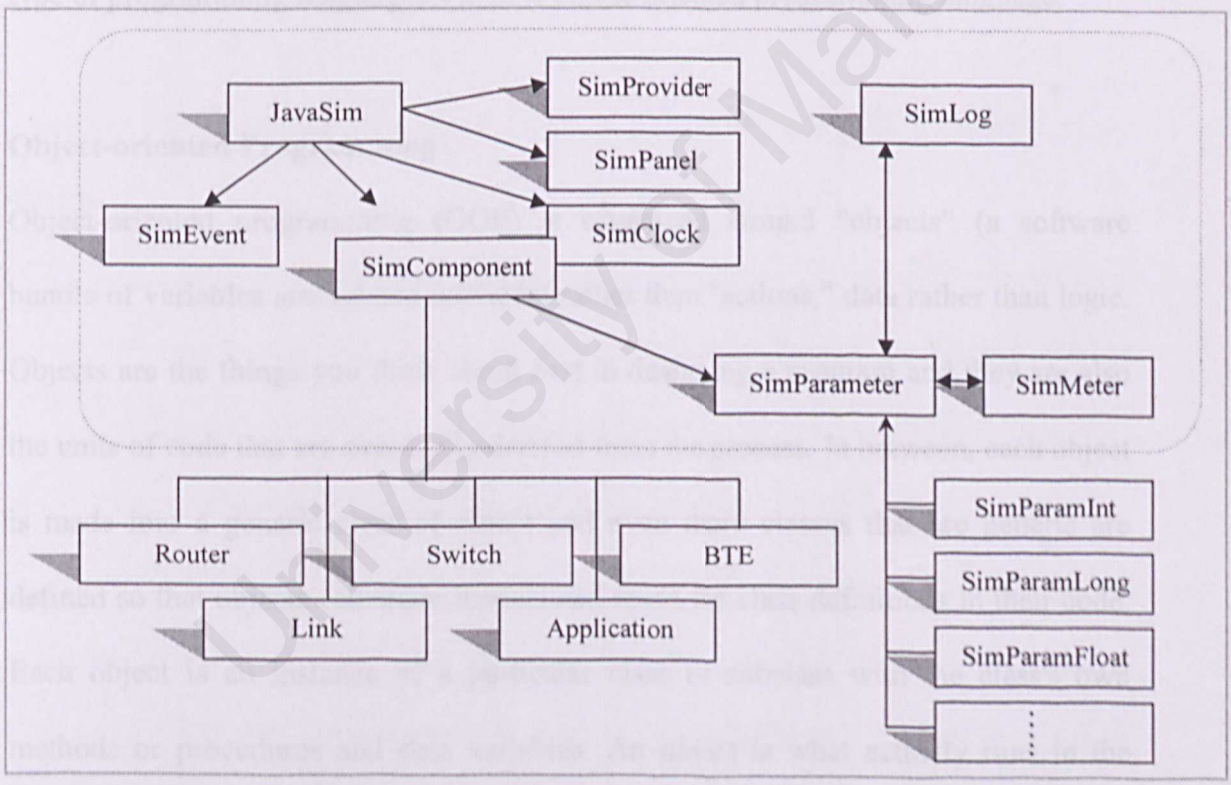


Figure 4. 4 UMJanetSim architecture

4.3 Programming Approach

Programming Language Selection

Before I start the implementation on network simulator, I must select to proper programming language. Therefore, it is a must to compare the advantages and disadvantages of programming languages that are possible to use to implement the component in network simulator. In the sub topic below, I have discuss about two type of programming languages which is object-oriented programming language.

Object-oriented Programming

Object-oriented programming (OOP) is organized around "objects" (a software bundle of variables and related methods) rather than "actions," data rather than logic. Objects are the things you think about first in designing a program and they are also the units of code that are eventually derived from the process. In between, each object is made into a generic class of object and even more classes that are generic are defined so that objects can share models and reuse the class definitions in their code. Each object is an instance of a particular class or subclass with the class's own methods or procedures and data variables. An object is what actually runs in the computer.

The first step in OOP is to identify all the objects you want to manipulate and how they relate to each other, an exercise often known as data modeling. Once you've

identified an object, you generalize it as a class of objects (think of Plato's concept of the "ideal" chair that stands for all chairs) and define the kind of data it contains and any logic sequences that can manipulate it. Each distinct logic sequence is known as a method. A real instance of a class is called (no surprise here) an "object" or, in some environments, an "instance of a class." The object or class instance is what you run in the computer. Its methods provide computer instructions and the class object characteristics provide relevant data. You communicate with objects - and they communicate with each other - with well-defined interfaces called *messages*.

The concepts and rules used in object-oriented programming provide these important benefits:

- The concept of a data class makes it possible to define subclasses of data objects that share some or all of the main class characteristics. Called inheritance, this property of OOP forces a more thorough data analysis, reduces development time, and ensures more accurate coding.
- Since a class defines only the data it needs to be concerned with, when an instance of that class (an object) is run, the code will not be able to accidentally access other program data. This characteristic of data hiding provides greater system security and avoids unintended data corruption.
- The definition of a class is reusable not only by the program for which it is initially created but also by other object-oriented programs (and, for this reason, can be more easily distributed for use in networks).

- The concept of data classes allows a programmer to create any new data type that is not already defined in the language itself.
- One of the first object-oriented computer languages was called Smalltalk. C++ and Java are the most popular object-oriented languages today. The Java programming language is designed especially for use in distributed applications on corporate networks and the Internet.

The following describe some advantages of object-oriented programming language.

- ***Simplicity:*** software objects model real world objects, so the complexity is reduced and the program structure is very clear.
- ***Modularity:*** each object forms a separate entity whose internal workings are decoupled from other parts of the system.
- ***Modifiability:*** it is easy to make minor changes in the data representation or procedures in an OO program. Changes inside a class do not affect any other part of a program, since the only public interface that the external world has to a class is through the use of methods.
- ***Extensibility:*** adding new features or responding to changing operating environment can be solved by introducing a few new objects and modifying some existing ones.
- ***Maintainability:*** objects can be maintained seperatelty, making locating and fixing problems easier.
- ***Reusability:*** objects can be reused in different programs. *(M. Melly, 1995)*

Java Programming

Java is a programming language expressly designed for use in the distributed environment of the Internet. It was designed to have the "look and feel" of the C++ language, but it is simpler to use than C++ and enforces an object-oriented programming model. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build a small application module or applet for use as part of a Web page. Applets make it possible for a Web page user to interact with the page.

The major characteristics of Java are:

- The programs created by Java are portable in a network. The source program is compiled into Java's byte code, which can be run anywhere in a network on a server or client that has a Java virtual machine. The Java virtual machine interprets the byte code into code that will run on the real computer hardware. This means that individual computer platform differences such as instruction lengths can be recognized and accommodated locally just as the program is being executed. Platform-specific versions of the program are no longer needed.
- The code is robust, here meaning that, unlike programs written in C++ and perhaps some other languages, the Java objects can contain no references to data external to themselves or other known objects. This ensures that an instruction cannot contain the address of data storage in another application or

in the operating system itself, either of which would cause the program and perhaps the operating system itself to terminate or "crash." The Java virtual machine makes a number of checks on each object to ensure integrity.

- Java is object-oriented, which means that, among other characteristics, an object can take advantage of being part of a class of objects and inherit code that is common to the class. Objects are thought of as "nouns" that a user might relate to rather than the traditional procedural "verbs." A method can be thought of as one of the object's capabilities or behaviors.
- In addition to being executed at the client rather than the server, a Java applet has other characteristics designed to make it run fast.
- Relative to C++, Java is easier to learn. *(Deitel, 1997)*

The major advantages of Java are:

- ***Simple:*** Java is simple for building a system that could be programmed easily without a lot of esoteric training and which leveraged today's standard practice.
- ***Object-oriented:*** Object-oriented facilities of Java are essentially those of C++, with extension from objective C for more dynamic method resolution.
- ***Network-Savvy:*** Java has an extensive library of routines from c opying easily with TCP/IP protocols like HTTP and FTP. This makes creating network connections much easier then is C or C++.

- ***Robust:*** Java objects can contain no references to data external or other known object. This ensures that an instruction do not contain the address of the data storage in another application or in the operating system itself to terminate or "crash". Garbage collection is another powerful feature that makes no chance for a Java program to corrupt the memory via a dangling pointer. The Java virtual machine makes a number of checks on each o ensure integrity.
- ***Secure:*** Java's run time system performs checking to unsure that programs transmitted over a network have not been tampered with. The code produced by the Java compiler is checked for validity, and the program is prevented from performing unauthorized actions.
- ***Architecture neutral:*** Networks are composed of variety of systems with a variety of CPU and operating system architecture. To enable a Java application to execute anywhere on the network, the compiler generates an architecture-neutral object file format – the compiled code is executable on many processors, given the presence of the Java runtime system.
- ***Portable:*** Java program is compiled into byte code instead of bit code. This feature make Java can be run anywhere in a network on a server or client.
- ***Interpreted:*** Java byte codes are translated on the fly to native machine instructions (interpreted) and not stored anywhere. Linking is a more

incremental and lightweight process, the development process can be much more rapid and exploratory.

- ***High performance:*** In addition to being executed at the client rather than the server, a Java applet has other characteristics design to speed up the execution.
- ***Multihtread:*** Java has built-in support for multitasking. A Jaa program may create any number of threads, which appear to execute in parallel.
- ***Dynamic:*** Java is more dynamic language than C or C++. It was design to adapt to an evolving environment.

C++ Programming

C++ is an object-oriented programming (OOP) language that is viewed by many as the best language for creating large-scale applications. C++ is a superset of the C language.

The C++ language facilities structured and disciplined approach to computer program design. C++ programs consist of pieces called classes and function. Programmer can program each piece that programmer need to form a c++ program. But most C++ programmers take advantage of the rich collections of existing classes and functions in the C++ standard library. C++ programs typically go through six phases to be executed. These are: edit, preprocess, compile, link, load, and execute. *(Deitel, 1997)*

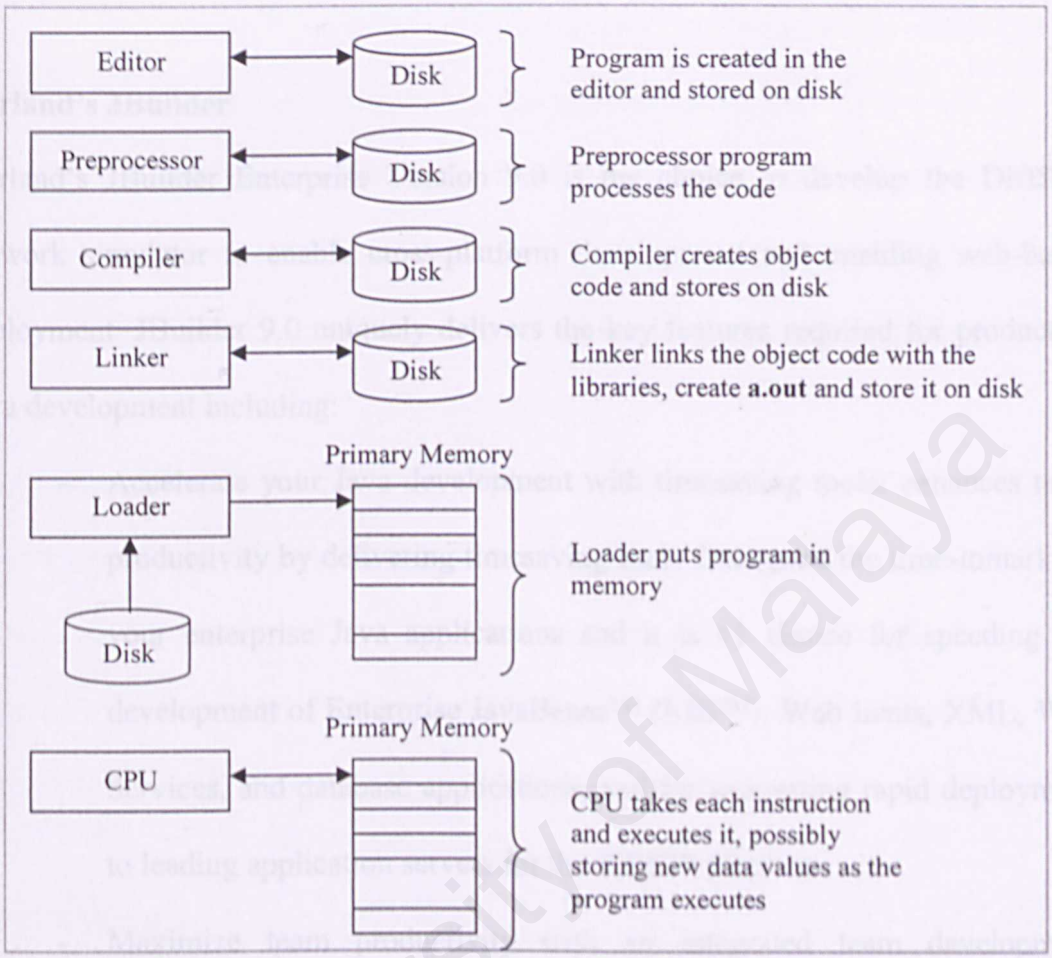


Figure 4. 5 Six phases of C++ programming language

4.4 *Software and Hardware Selection*

There are many types of Java programming tools available in market. One can develop Java programming using pure Java SDK without the supporting of integrated development environment (IDE), or just selects tools like Microsoft J++, Borland's JBuilder, Visual Age, Visual J++ or JCreator.

Borland's JBuilder

Borland's JBuilder Enterprise Version 9.0 is the choice to develop the DiffServ network simulator to enable cross-platform development and enabling web-based deployment. JBuilder 9.0 uniquely delivers the key features required for productive Java development including:

- Accelerate your Java development with timesaving tools: enhances team productivity by delivering timesaving tools that speed the time-tomarket f your enterprise Java applications and it is #1 choice for speeding the development of Enterprise JavaBeans™ (EJB™), Web lients, XML, Web Services, and database applications, and for supporting rapid deployment to leading application servers for the J2EE™ platform.
- Maximize team productivity with an integrated team development environment: JBuilder is engineered to enable Java development teams to create reliable, enterprise-class applications — fast.
- Stay focused on development with reliable tools integration: JBuilder development environment gives developers access to all stages of the application development lifecycle — from design, development, debugging and testing, to deployment and management.
- Reduce the risk and cost of development — all the way through deployment: JBuilder gives you the freedom to choose your preferred

development platforms and your version control system, releasing your business from the escalating expenses associated with vendor lock-in.

- Leverage investments in existing projects and manage change successfully: JBuilder technologies give a flexible platform that is sufficiently extensible to meet the ever-evolving needs of Java enterprise development environments.

"We chose JBuilder for all current and future development because it allows us to meet two key business objectives: to use technology that delivers the highest levels of services to our customers while also employing best practices to provide low process costs."

[Stephen Reyes, Senior Manager, Web Development, ProBusiness]

Xinox's JCreator

JCreator is a powerful IDE for Java. JCreator provides the user with a wide range of functionality such as : Project management, project templates, code-completion, debugger interface, editor with syntax highlighting, wizards and a fully customizable user interface. JCreator is written entirely in C++, which makes it fast and efficient compared to the Java based editors/IDEs. With JCreator you can directly compile or run your Java program without activating the main document first. JCreator will automatically find the file with the main method or the html file holding the java applet, then it will start the appropriate tool. JCreator is a powerful IDE for Java™ technologies that provides more power at your fingertips than all the ordinary IDEs

combined. The following features cause many software developer choose it as their software development tool:

- Manage projects with ease in the interface that is much like Microsoft® Visual Studio®.
- Define your own colour schemes for unlimited ways to organize your code.
- Unlike most IDEs, JCreator wraps around your existing projects and allows you to use different JDK profiles.
- Get down to writing code quickly with our project templates.
- Our class browser makes viewing your project a breeze.
- Debug with an easy, intuitive interface. No need for silly DOS prompts!
- Walk through our wizards and cut to the chase of writing your project, quickly and easily.
- You don't have to spend valuable time on Classpath configuration—JCreator does it all for you.
- Customize our user interface the way that you like it.
- Set up your own run-time environments to run your application as an applet, in a JUnit environment or in a DOS window.
- JCreator has lower system requirements, yet faster speed, than all those other IDEs.

Platform

Refer to Table 2.6, most of the currently available simulators are running on UNIX platforms. With the rapid development on Windows system, the processing speed of a PC is comparable with a UNIX system and the price of a PC that runs on Windows platform is lower compare to a UNIX workstation. Therefore, the Windows platform becomes dominant at the recent days. Although Windows platform becomes more common, The UNIX platform can't be neglected.

A network simulator that is cross-platform is more important as it is supported by most of the platform, for example UNIX or Windows

4.5 Functional Requirement

The main purpose of the analysis for functional requirement is to ensure the developed simulator will satisfy user's needs and requirements.

The minimum functional requirements of a network simulator are as follows:

- The simulator will has a user interface with menu bar to allow user to choose the necessary features and steps.
- The simulator will allow user to configure the parameters
- The router allow user to set the type of marker to be choosen

- The simulator will run based on the parameter setting and network topology
- This simulator will support two type of three colour marker

4.6 *Non-Functional Requirement*

Reliability: The system should be reliable in performing its simulation function and network operations. For example, whenever a button is clicked, the system should be able to perform some functionality or generate some message or animation to inform the user what is happening

Usability: The system should be user friendly. User must be able to learn how to use the system in the shortest period. It will enhance and support rather than limit or restrict the understanding of DiffServ Network. Human interfaces need to be intuitive and consistent with the user knowledge in order to let them gain some knowledge through the simulator.

Manageability: The modules within the system should be easy to manage. This will make the maintaining, enhancement work simpler, and not time consumed. The system should not cause any damages to the current simulator after a new component has been added. The system should be designed systematically so that the effort required to locate and fix an error in the system is minimum.

Flexibility: The system has its capabilities to make advantages of new technologies, resource and in fast changing environment. The system should be able to implement in a changing environment of platform.

Correctness: The final application must meet the objective, specification and requirement of the users.

4.7 Summary

This chapter covers the major analysis on the existing UMJanetsim has been analyzed from the user interface to the system architecture and the API. The overall architecture of UMJanetsim network simulator is analyzed in order to figure out how the new component should be integrated into the existing simulator without affecting the main features and performances.

UMJanetSim is an object oriented approach system, and the java programming language was the ideal choice. This is because it is object oriented and it contains the built in support for multithreaded programming. Moreover, Java is robust compared to C++ as it has no pointer references to other data. In addition, Java was selected due to its portability, which makes it platform independent.The preferred programming tools that has been chosen is Xinox's JCreator due to the wide range of functionality.

CHAPTER 5 SYSTEM DESIGN

This chapter is about the stage of system design. After the stage of system analysis, the implementation of the three colour marker can be design according to the system's requirement. The object-oriented methodology will be used at the implementation phase of the development process through the java programming language.

5.1 UMJanetSim Design Overview

The overview of UMJanetSim design is important before the design of the new implementation in it. The architecture of the UMJanetsim is as Figure 5.1.

JavaSim is the main simulation object. There will be only one instance of the Sim object throughout the simulation. Anyone with a reference to this instance can make use of the following services:

```
public long now();
        //returns current simulation time (in tick)

public java.util.List getSimComponents();
        //returns a list of all existing SimComponent
```

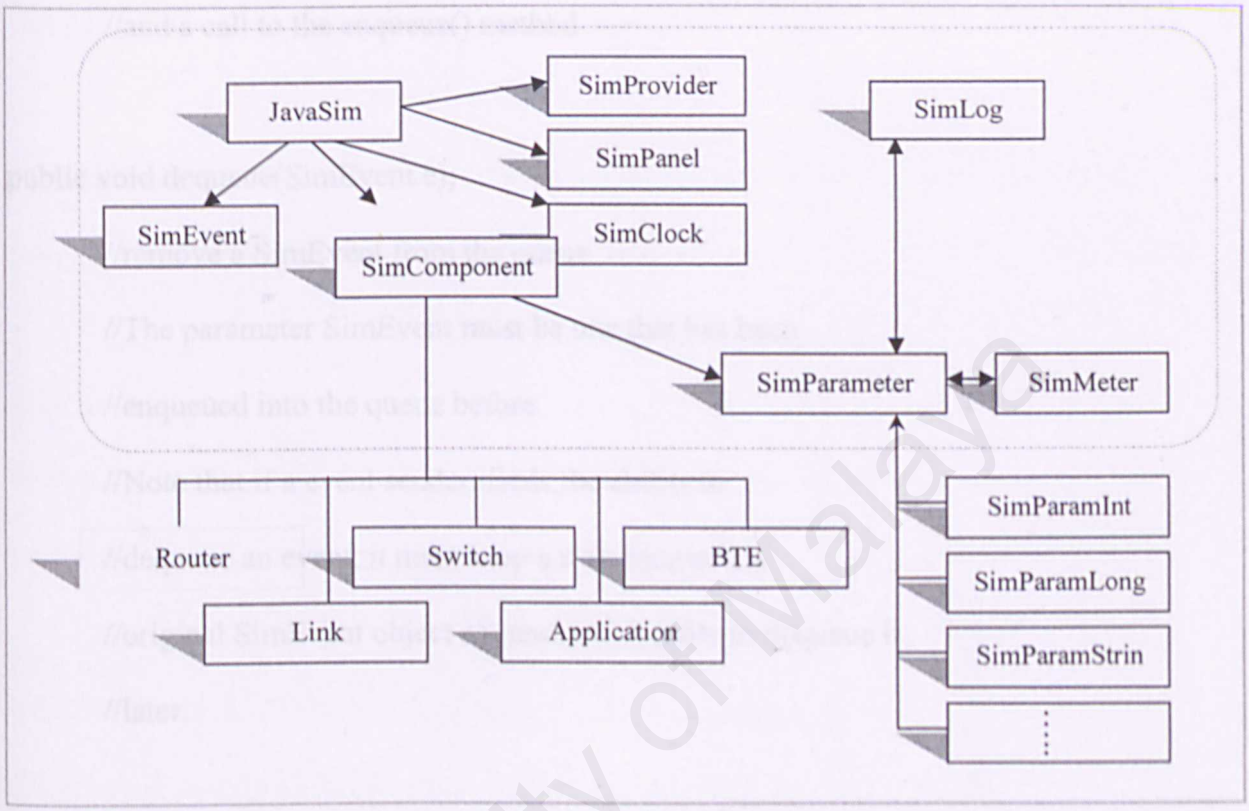


Figure 5. 1 UMJanetSim architecture

```
public boolean isRunning();
        //to check whether the simulation is running.
        //Complex GUI manipulation should not be done when
        //the simulation is running.

public void enqueue(SimEvent e);
        //the single most important method to remember!
        //Every communication (message exchange) between
```

```
        //any components must involve creation of a SimEvent
        //and a call to the enqueue() method

public void dequeue(SimEvent e);
        //remove a SimEvent from the queue.
        //The parameter SimEvent must be one that has been
        //enqueued into the queue before.
        //Note that if a event sender needs the ability to
        //dequeue an event, it must keep a reference of the
        //original SimEvent object somewhere in order to dequeue it
        //later.

public void dequeue_by_comp_and_type(SimListener src, SimListener dest, int type);
        //dequeue all events in the event queue the match the
        //provided source, destination, and event type (must
        //  match all). The parameter src and/or dest can be null
        //  to act as a wildcard (match all).
```

SimComponent

The SimComponent is the most important class to understand in the simulator in order to development new components. Every network component in the simulation MUST inherit (i.e. extend) SimComponent. The actual behavior of a component is

achieved by overriding appropriate methods of SimComponent in order to react to simulation events and other simulation operations.

The SimComponent class is within the *janetsim* package, but all components derived from it should be in the *janetsim.component* package.

SimClock

SimClock rovides a set of time translation functions (all static) for translation between tick and actual time (microseconds, seconds etc.).

```
public static double Tick2Sec(long tick); //ticks to seconds
public static double Tick2MSec(long tick); //ticks to msecs
public static double Tick2USec(long tick); //ticks to usecs
public static long Sec2Tick(double sec); //seconds to ticks
public static long MSec2Tick(double msec); //msecs to ticks
public static long USec2Tick(double usec); //usecs to ticks
public static String getClockString(long tick);
        //get a formatted representation of a given time
```

SimParameter

Every SimComponent can have internal parameters (not shown/accessible by users) or external parameters (shown/accessible by users). All external parameters MUST

inherit SimParameter. By extending SimParameter, one obtains parameter logging and meter display features automatically.

```
public SimParameter(String aName,SimComponent comp, long creationTick,
                    boolean isLoggable);
    //Any parameter that inherits SimParameter should provide a
    //constructor that includes at least the above four
    //parameters, which in turn, should be passed directly to
    //      super(aName,comp,creationTick,isLoggable);
    //as the first statement of the constructor. The constructor
    //can accept additional parameters if needed (especially
    //the actual value of the parameter, which is not included
    //in the basic constructor in SimParameter)
    //The parameters are:
    //      aName – name of the parameter
    //comp – parameter owner
    //creationTick – time when the parameter is created
    //isLoggable – whether the parameter can be logged in the
    //                    log file
```

5.2 *Three Colour Marker Design*

TCM is designed to implement in the IPRouter class. It will be built on top of javasim package. There are two types of marker will be implemented in the router. The user is allowed to choose either one to simulate the network. The marker will firstly monitor and meter the traffic from source from TCP application, UDP application, UDP CBR, CBR application and VBR application. Next, the meter will determine how the traffic will be marked.

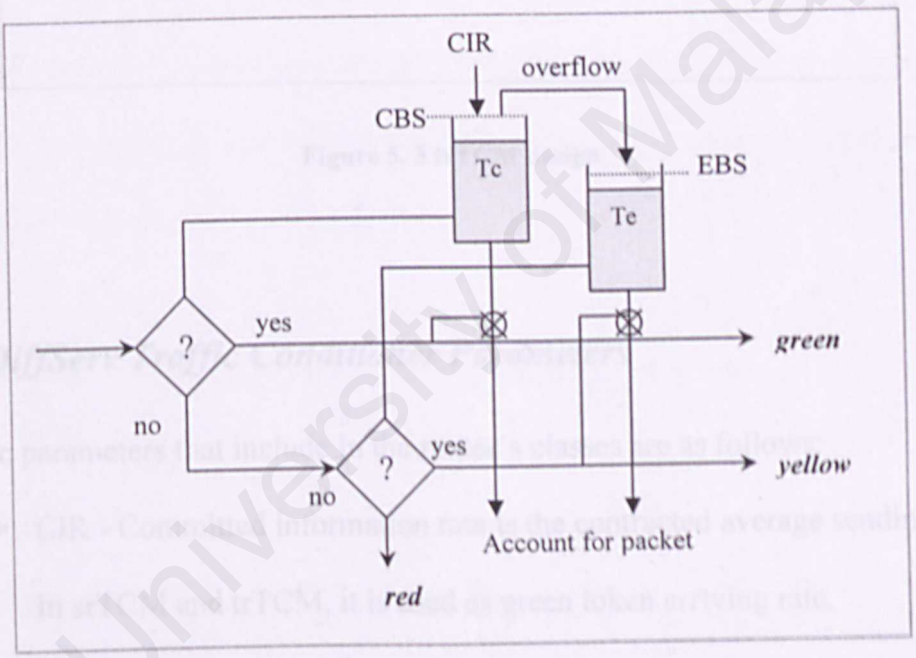


Figure 5. 2 srTCM Design

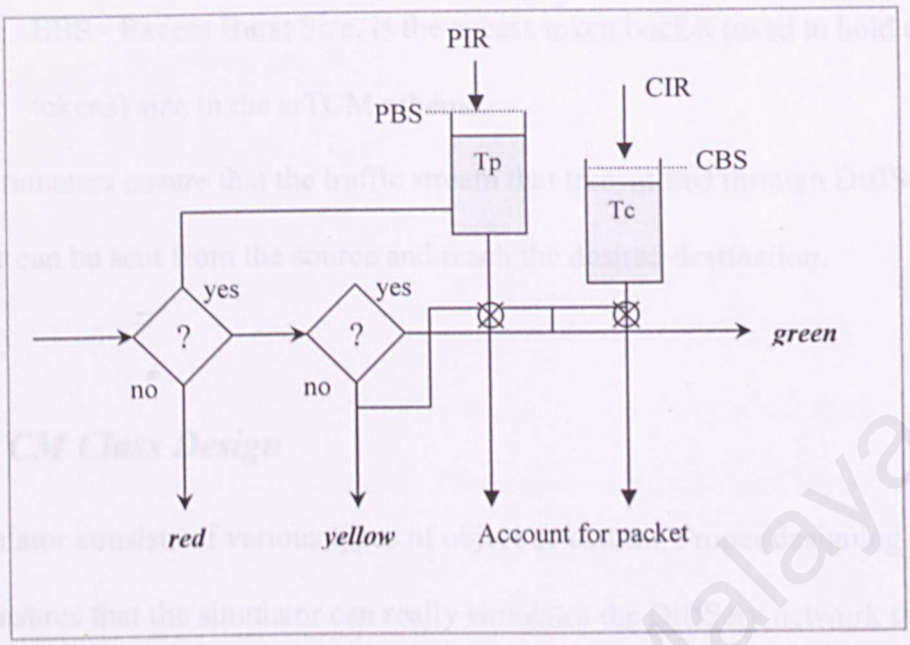


Figure 5. 3 trTCM design

5.3 *DiffServ Traffic Conditioner Parameters*

The basic parameters that include in the router's classes are as follows:

- CIR - Committed information rate is the contracted average sending rate. In srTCM and trTCM, it is used as green token arriving rate.
- PIR - Peak information rate is the maximum contracted sending rate. In trTCM marking scheme, it is used as the yellow token bucket size.
- CBS - Committed Burst Size, is the contracted traffic burst size. In srTCM and trTCM, it is used as the green token bucket size.
- PBS - Peak Burst Size, is the yellow token bucket size in the trTCM scheme.

- EBS - Excess Burst Size, is the excess token bucket (used to hold excess tokens) size in the srTCM scheme.

These parameters ensure that the traffic stream that transmitted through DiffServ simulator can be sent from the source and reach the desired destination.

5.4 TCM Class Design

The simulator consists of various types of object or classes. Proper designing of these classes ensures that the simulator can really simulates the DiffServ network that imitates the real network. The simulator is builds from many classes like IPRouter class, RIPRouter class, RIP class, etc.

The classes that will be implemented in UMJanetsim are as below:

```
private int checkMeter()
    //check the type of marker, either srTCM or trTCM
    //if srTCM, get parameter: CIR, CBS and EBS
    //use the algorithm of srTCM according to the flow chart
    //if trTCM, get parameter: CIR, PIR, CBS, PBS
    //use the algorithm of trTCM according to the flow chart
    //return value '0' if the meter determine to mark the packet as green
    //return value '1' if the meter determine to mark the packet as yellow
```

```
        //return value '2' if the meter determine to mark the packet as red

private int checkDropper()

        //check whether the packet is been drop or not

        //call method checkMeter(), if the value return by checkMeter()=2, then will
        //      drop the packet. If the value return by checkMeter()=0, then the packet
        //      will be pass to next node. If the value reutrn by checkMeter()=1, then
        //      will call class checkYellow().


private int checkYellow()

        //will check the packet again, to ensure the packet to be drop or not. If the
        //      packet is a high drop precedence packet, return value '2' to
        //      checkDropper() so that the dropper will drop the packet. If not, then
        //      will return value '0', so that the packet can be pass to next node


private int updateTokenCount()

        //this method is to update the token count for each token bucket.
        //this method will be called by checkMeter() to make sure the toke count is the
        //      latest token count before the packet can be meter to determine how it
        //      will be mark.
```

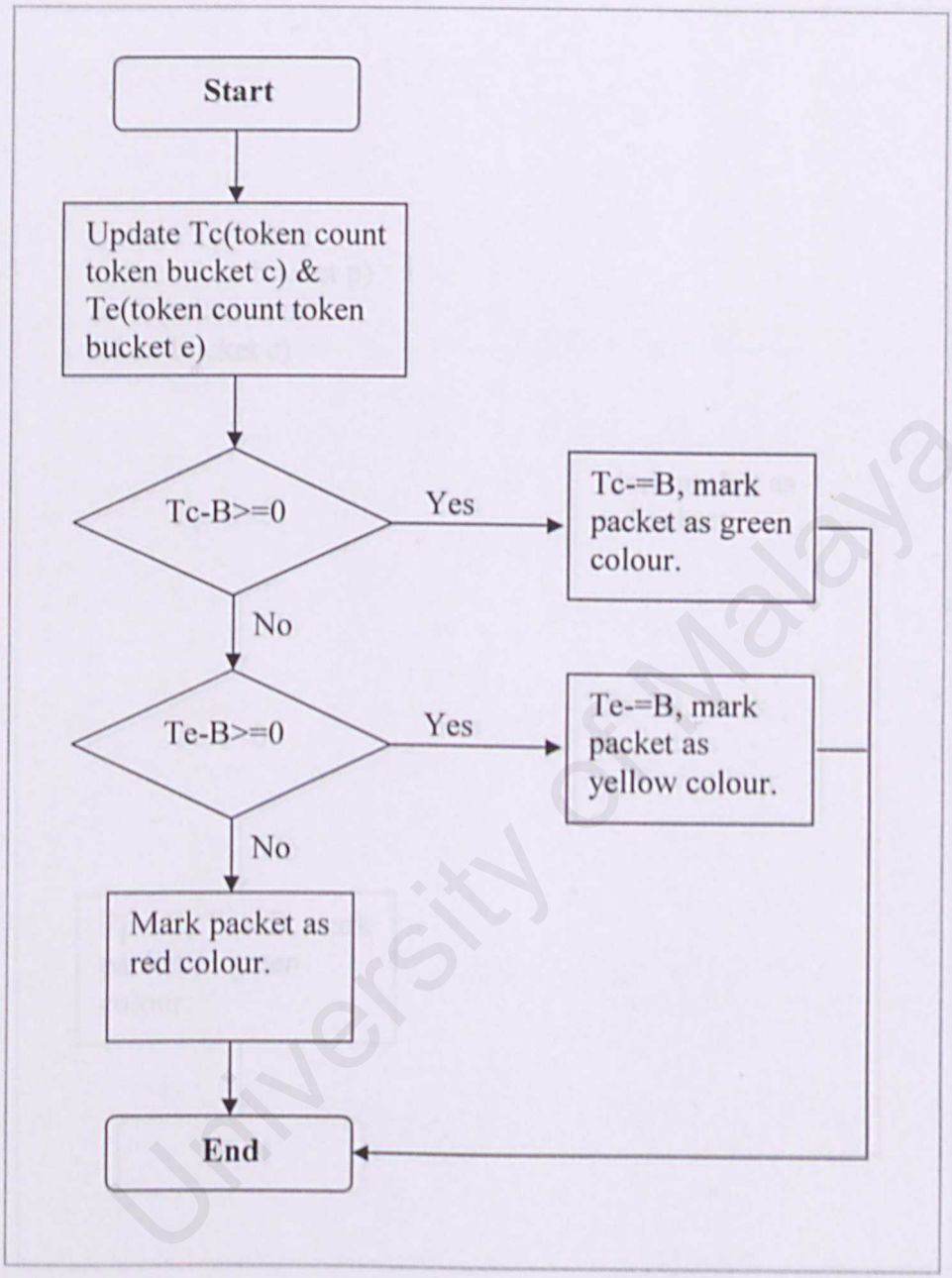


Figure 5. 4 Flow chart for srTCM meter

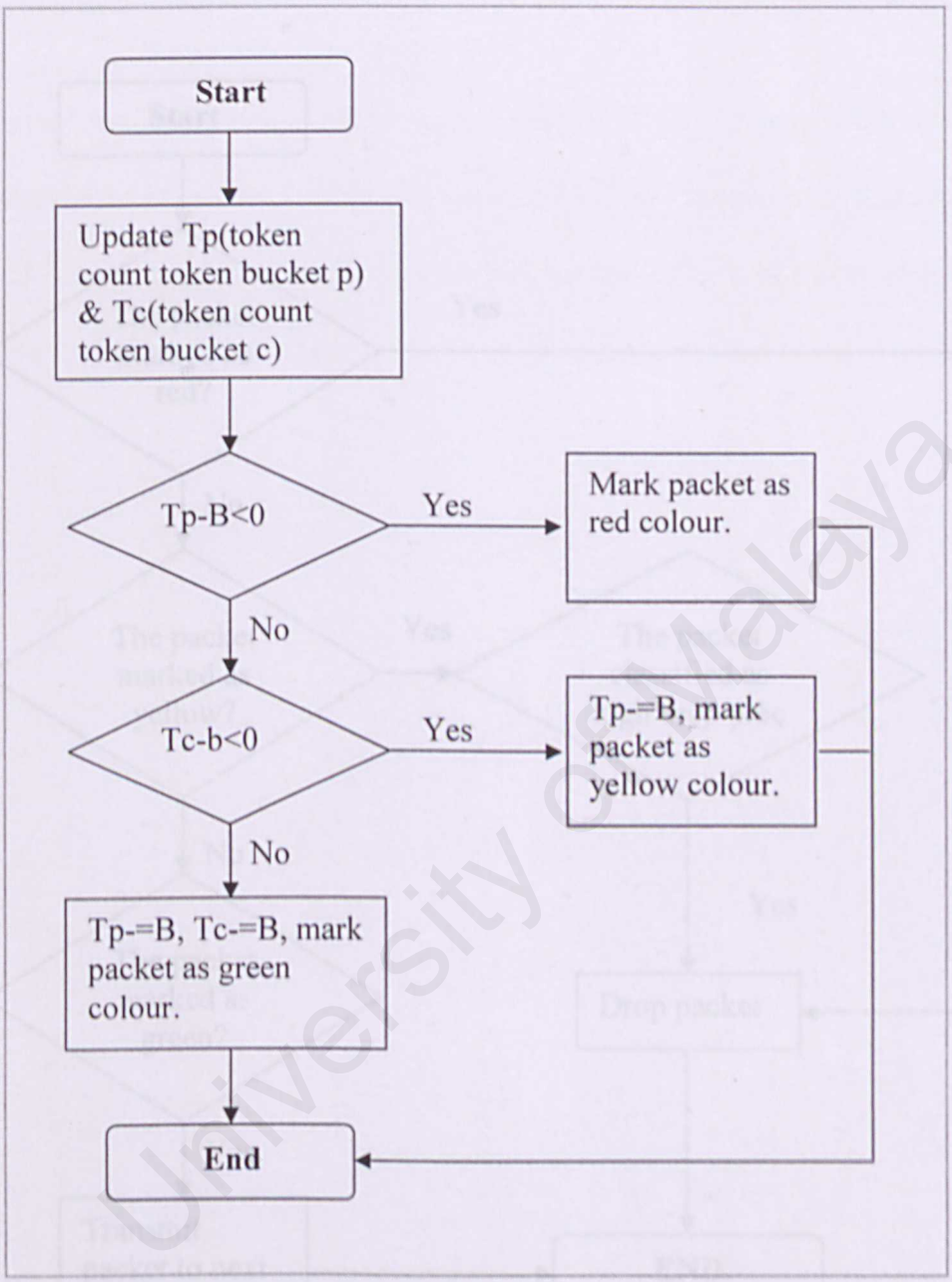


Figure 5. 5 Flow chart for trTCM meter

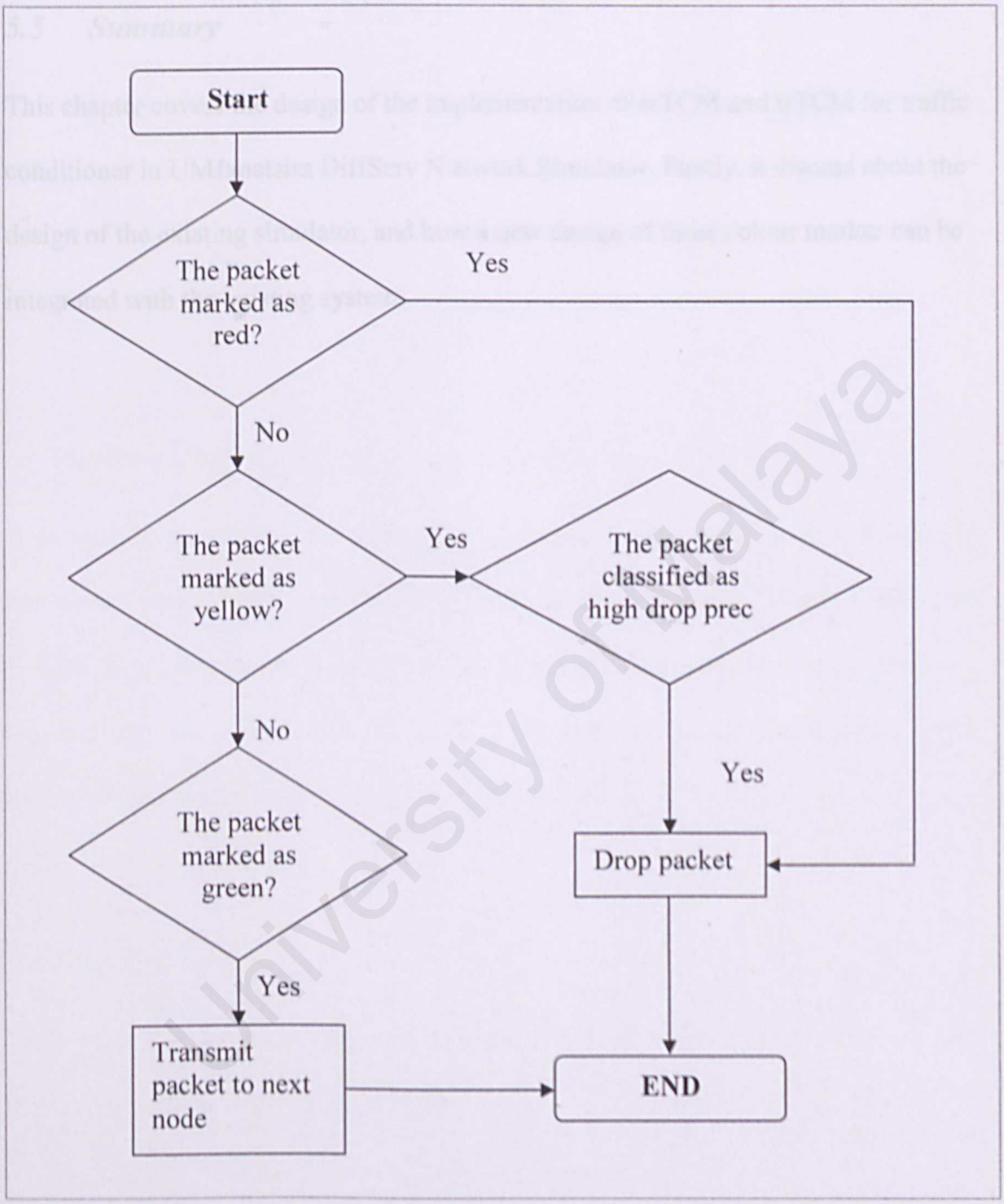


Figure 5. 6 Flow chart for TCM dropper

5.5 Summary

This chapter covers the design of the implementation of srTCM and trTCM for traffic conditioner in UMJanetsim DiffServ N etwork Simulator. Firstly, it discuss about the design of the existing simulator, and how a new design of three colour marker can be integrated with the existing system

CHAPTER 6 IMPLEMENTATION

This chapter presents the implementation phase of the three colour marker in diffserv network simulator. The relevant classes and the new method together with the attributes will be shown. Each attribute as well as methods contained within the classes will be explained in detail.

6.1 Implementation

The UMJanetsim simulator main package is janetsim. All classes that have to be modify to implement three colour marker is in the janetsim package. These classes are UDP_CBR.java, IPPacket.java, EtherFrame.java, and IPRouter.java. Besides, a new class named TrafficProfile is added in for traffic conditioner of the simulator. Next, I will describe attributes within classes of classes listed above.

6.2 Traffic Profile

The only class that has been add in UMJanetSim is TrafficProfile.java. There are four attributes in this class, which are as below:

```
class TrafficProfile implements java.io.Serializable {
        double peakrate; //Mbps
        double meanrate; //Mbps
        int burstsize; //bytes
        int burstsize2; //bytes
```

```
}
```

This class will be called in UDP_CBR.java or IPRouter when the traffic profile is enabled.

6.3 *UDP_CBR Application*

In Udp Cbr, a SimParamBool to enable traffic profile, two SimParamDouble . and two SimParamInt has been added. Two SimParamDouble are peak rate and committed rate, which committed rate is for both srTCM and trTCM and peak rate is only for trTCM.

When there is a neighbor added to the udp cbr, two SimParamLong(cn_total_sent and cn_tc_drop) will be added in it. cn_total_sent is to show the total frames that will be sent by the particular application, while cn_tc_drop will shows the total frames dropped in edge router.

Method compInfo is added, so that when there is frame dropped in edge router it will be call to increase the counter of drop, cn_tc_drop.

```
public Object compInfo(int infoid,SimComponent src,Object paramlist) {
        switch(infoid) {
                case UDP_CBR.CI_FRAME_TC_DROP:
                        cn_tc_drop.setValue(cn_tc_drop.getValue()+1);
                        cn_tc_drop.update(theSim.now());
```

```
            return null;

        }
```

6.4 *IPPakcet and EtherFrame*

In the implementation of three colour marker, both IPpacket and EtherFrame has added the same coding. In both classes, the object of TrafficProfile named t_profile is created. Next, the constructor with arguments is added.

```
IPPacket(SimComponent thesender,long tick) {
        sender=thesender;
        sendingTick=tick;
}
EtherFrame(SimComponent thesender,long tick) {
        sender=thesender;
        sendingTick=tick;

}
```

The same modification is done on these classes. When there are data to be sent to destination, IPPacket will be queue in TheSim. After the IPPacket is received in router, then only it will be transform into frame.

6.5 IPRouter

IPRouter is the most important class in the implementation of the three colour marker in the UMJanetSim. The method of traffic conditioner is added in this class. Theoretically, the traffic conditioner is in the edge router, therefore IPRouter is the most suitable class to be chosen for this implementation.

There are few types of parameters have been added in IPRouter.java. The list of parameters is shown below:

```
protected SimParamInt sw_frames_received;
protected SimParamDouble sw_dropped_tc;
protected SimParamInt sw_dropped;
protected SimParamInt sw_markRed;
protected SimParamInt sw_yellow;
protected SimParamInt sw_yellow_dropped;
protected SimParamInt sw_green;
```

Table 6.1 is the summary description of each parameter.

Parameter	Description
sw_frames_received	To show the total number of frames received in that particular router
sw_dropped_tc	To show total number of frames dropped by traffic conditioner which equal to sw_markRed + sw_yellow_dropped

sw_markRed	To show total number of frames have been marked as red colour by traffic conditioner
sw_yellow	To show total number of frames have been marked as yellow colour by traffic conditioner
sw_yellow_dropped	To show total number of frames dropped that have been marked as yellow colour by traffic conditioner
sw_green	To show total number of frames have been marked as green colour by traffic conditioner

Table 6 1 Parameters in IPRouter

Other than the parameter listed above, there are also one SimParamBool (sw_tc), one SimParamIntTag (sw_marker), two SimParamDouble (sw_peakrate, sw_meanrate) and two SimParamInt (sw_burstsize, sw_burstsize2) have been added to allow user to input. To simulate network with the traffic conditioner of three colour marker, the sw_tc must be enabled. Next, the user is allowed to choose either srTCM or trTCM from sw_marker.

Refer to 6.3, these parameters are added in udp cbr application as well. The redundant of the parameters is to allow to type of testing. If there are few applications connected to the particular router, and the sw_tc is enabled in this router, the global testing with global traffic profile in it can be done. Besides, the user can choose to test the application

individually with their traffic profile, the user have to enable the traffic profile in that application.

In the class of send_etherframe, if the sw_tc.gevalue=true, the value of packet.t_profile will be called to t_profile in router. Before the method to queue the frame to be sent out from router, the new method sw_tc_perform will be called to determine the frame to be drop or queue in the output. If sw_tc_perform return true, frame will be equal to null, which mean the frame has been dropped. Oppositely, the frame will be ready to send to the next router.

6.5.1 sw_tc_perform

The main function of sw_tc_perform is to inform send_etherframe whether the traffic conditioner determines to drop the frame or not. The latest token count will be keep in ProfileRecord. The class of ProfileRecord is implemented as follow:

```
private class ProfileRecord implements java.io.Serializable {
        int token_count1, token_count2;
        long lastpackettime;
}
```

The token_count1 is the total number of token in first token bucket that has the committed burst size and the token_count2 is the latest number of token in the token bucket with peak or excess burst size.

Next, the method checkCoforming will be called to check the conform_level. The implementation of checkConforming will be explained in details in 6.5.2. The value of

conform_level return from checkConforming method will determine the frame either to be sent to next node or to be dropped immediately.

```
if(conform_level<-1) tc_drop=true;
if(conform_level==-1){
        if((voport.outQ_size.getValue())/sw_oqsize.getValue()*100>90) {
                sw_yellow_dropped.setValue(sw_yellow_dropped.getValue()+1,
                  theSim.now(),sw_log_factor.getValue());
                sw_yellow_dropped.update(theSim.now());
                tc_drop=true;
        }
}
```

If conform_level<-1, tc_drop=true. As the conform_level=-1, it will check the output queue size, if the output queue is 90% utilized, the only tc_drop will be equal to true.

In the final part of this method, it will check if the tc_drop is true, then the variable num_dropped will increase one. Other than that, it will also notify the application sender that traffic conditioner has drop the frame as mentioned in chapter 6.3.

```
if(frame.sender!=null && frame.sender.getCompClass().equals("Application")) {
        Object [] paramlist=new Object[1];
        paramlist[0]=frame;
        frame.sender.compInfo(UDP_CBR.CI_FRAME_TC_DROP,this,
                paramlist);
}
```

6.5.2 checkConforming

This method is developed to check and mark the frames. The switch control structure is used to divide the algorithm of srTCM and trTCM.

```
switch(sw_marker.getValue()) {
        case 0: //srTCM
        case 1: //trTCM
}
```

srTCM

The first step in the marker of traffic conditioner is to check and update the token count for each token bucket. The algorithm to update the token count is as below:

```
bytes=(int)(frame.t_profile.meanrate*SimClock.Tick2USec(
        now-prof.lastpackettime)/8);
if(bytes>0 && (prof.token_count1<frame.t_profile.burstsize)) {
        int diff=frame.t_profile.burstsize-prof.token_count1;
        if(diff>=bytes) {
        prof.token_count1+=bytes;
        bytes=0;
        }
        else {
                prof.token_count1=frame.t_profile.burstsize;
                bytes-=diff;
```

```
    }
}
if(bytes>0 && (prof.token_count2<frame.t_profile.burstsize2)) {
    prof.token_count2=Math.min(prof.token_count2+bytes,frame.t_profile.burstsize2);
}
prof.lastpackettime=now;
```

Byte is the size of token ready to be refill in the token bucket. If the first bucket is overloaded, the token ready will be refill in second bucket, else it will refill in the first bucket.

Next, the marker will start its role to mark the frame. The algorithm to determine how the frame marked srTCM is as below:

```
if((prof.token_count1 - frame.len) >= 0) {
        prof.token_count1-=frame.len;
        return 0;  //marked as green
}
if((prof.token_count2 - frame.len) >= 0) {
        prof.token_count2-=frame.len;
        return -1;  //marked as yellow
}
//If not the above condition then will reutnr -2
return -2;  //marked as red
```

In the marker, firstly it will check the frame length and the first token count. If the token count is enough for frame length, then it will be marked as green. In the other way, it will then check with then second token count. If the second token count if more than frame length, then it will be marked as yellow colour. If neither of the above condition, then the frame will be marked as red colour which it will be drop in traffic conditioner.

trTCM

As in srTCM, before the comparison of token count, the token buckets will update their token count. The algorithm to update the token count are as below:

```
bytes=(int)(frame.t_profile.meanrate*SimClock.Tick2USec(now-prof.lastpackettime)/8);
if(bytes>0 && (prof.token_count1<frame.t_profile.burstsize)) {
        prof.token_count1=Math.min(prof.token_count1+bytes,frame.t_profile.burstsize);
}
bytes=(int)(frame.t_profile.peakrate*SimClock.Tick2USec(now-prof.lastpackettime)/8);
if(bytes>0 && (prof.token_count2<frame.t_profile.burstsize2)) {
        prof.token_count2=Math.min(prof.token_count2+bytes,frame.t_profile.burstsize2);
}
prof.lastpackettime=now;
```

Each of the token buckets has its own token rate, and the tokens are refilled respectively. The first token bucket refills the token by mean rate, while the second token bucket is by peak rate.

Next, is the algorithm of trTCM to mark the frames:

```
if(prof.token_count2<0) {
        return -2;  //marked as red
}
if(prof.token_count1<0) {
        prof.token_count2-=frame.len;
        return -1;  //marked as yellow
}
If not the above condition then will reutnr 0
prof.token_count2-=frame.len;
prof.token_count1-=frame.len;
return 0;
```

The marker will check the token count in the second token bucket first. If there is no token in the second bucket, then the frame will be marked as red. Oppositely, it will check the first token bucket, if there is no token in it then the frame will be marked as yellow and the number of token count in second token bucket will decrease by the size of the frame. However, if both of the token buckets are not empty, then the token count in both buckets will be decrease by the size of the frame and the frame will be marked as green colour.

6.6 *Summary*

This chapter has cover the way to implement two types of three colour marker in UMJanetSim, which are srTCM and trTCM. Generally, there are only few original

classes have been modify in order to suite the simulator to perform traffic conditioner. Five sub-topics at above describe how each classes has been modify and how the new method has been added in its. Other than that, there is also new class has been develop in this simulator.

CHAPTER 7 TESTING

Chapter 7 discusses the testing phases that had been done after the implementation of three colour marker. This phase is divided into three stages. The first stage is the component testing, which is to ensure that the input from the GUI is work perfectly. Next, each module will be test respectively on the functionality to perform traffic conditioner. Finally, is to test the whole system to make sure the integration of the new module work correctly without interruption on the existing module.

The simple topology that has been used for component testing, module testing and system testing is as below:

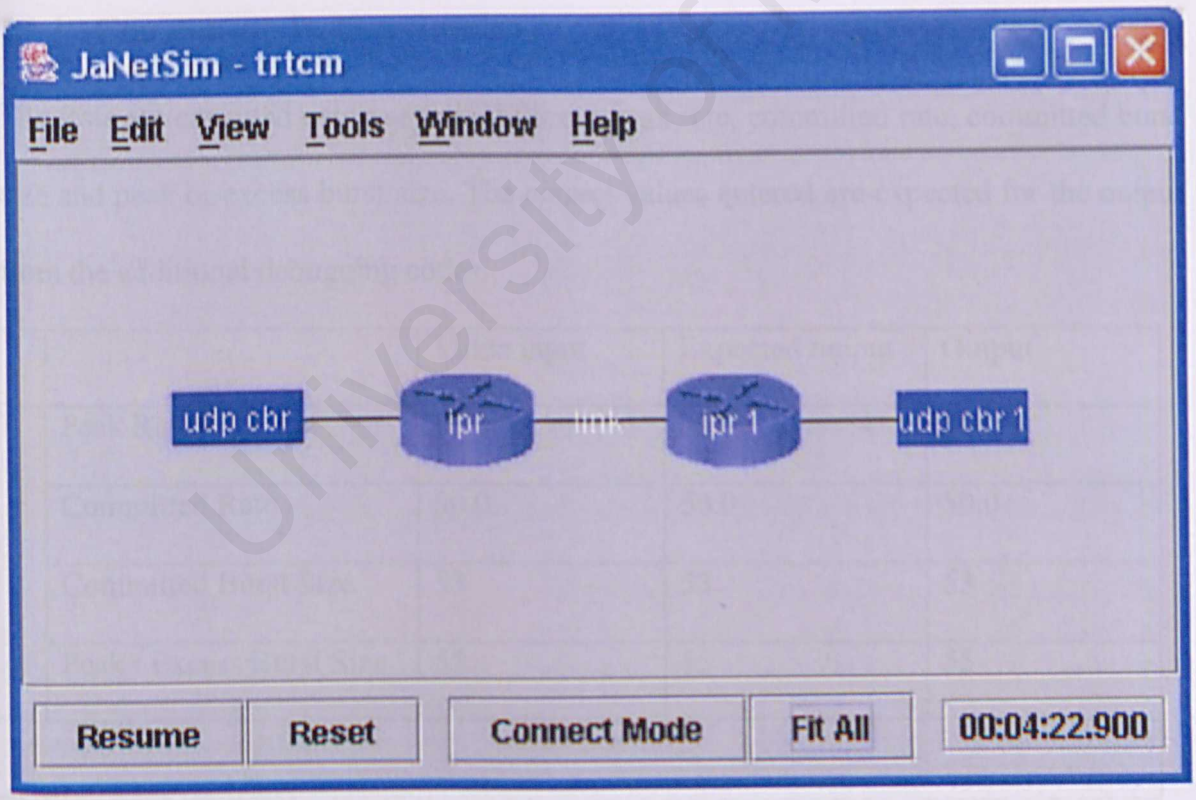


Figure 7 1 Topology for Component Testing, Module Testing & System Testing

7.1 Component Testing

This test is used to check the correct value of traffic profile's parameters entered into UDP_CBR and IPRouter. The System.out.println statement is used for testing purpose. The example of testing the value of traffic profiles in IPRouter.

```
System.out.println("Peak Rate (Mbps) >>" +sw_peakrate.getValue());
System.out.println("Committed Rate (Mbps) >>" + sw_meanrate.getValue());
System.out.println("Committed Burst Size (bytes)>>" + sw_burstsize.getValue());
System.out.println("Peak/Excess Burst Size (bytes) >>" + sw_burstsize2.getValue());
```

7.1.1 Component Testing Result

The tests are executed using several different peak rate, committed rate, committed burst size and peak or excess burst size. The correct values entered are expected for the output from the additional debugging codes.

		Value input	Expected output	Output
1	Peak Rate	80.0	80.0	80.0
	Committed Rate	50.0	50.0	50.0
	Committed Burst Size	53	53	53
	Peak / Excess Burst Size	55	55	55
2	Peak Rate	75.5	75.5	75.5
	Committed Rate	55.5	55.5	55.5
	Committed Burst Size	60	60	60

	Peak / Excess Burst Size	80	80	80
3	Peak Rate	125.5	125.5	125.5
	Committed Rate	105.3	105.3	105.3
	Committed Burst Size	73	73	73
	Peak / Excess Burst Size	4	4	4

Table 7 1 Results of Component Testing

7.2 Module Testing

The module testing is to check the functionalities of sw_tc_perform and checkConforming. The sw_tc_perform and checkConforming testing is to verifiy that:

I. It get the correct value of traffic profile from UDP_CBR

II. The ProfileRecord keep the traffic profile correctly

III. The conform_level get the value return from checkConforming

IV. The tc_drop control structure work properly

To check the frame.t_profile and ProfileRecord in IPRouter, I use the System.out.prinln as in the component testing. Sw_tc_perform and checkConforming is two method that relate to each other, the method of checkConforming will be called in sw_tc_perform to get the value for conform_level. Bottom-up method has been used in the module testing where the testing started from checkConforming.

7.2.1 Property Setting

There are two type of testing has been done. Table 7.2 show the property setting for srTCM testing and table 7.3 shows the property setting for trTCM. There are three test in each type of testing. The first part in each table is the property setting in application sender, while the second part is in the edge router.

	Test 1	Test 2	Test 3
Bit Rate	80	80	80
Packet Size	53	53	53
Number Sent	0.001696	0.001696	0.001696
Total Frame Sent	4	4	4
Committed Rate	50.0	50.0	50.0
Committed Burst	53	53	50
Prak/ Excess Burst Size	50	53	53

Table 7 2 Property Setting for srTCM

	Test 1	Test 2	Test 3
Bit Rate	80	80	80
Packet Size	53	53	53
Number Sent	0.001696	0.001696	0.001696
Total Frame Sent	4	4	4
Peak Rate	60.0	90.0	90.0
Committed Rate	50.0	50.0	85.0
Committed Burst	10	10	10
Prak/ Excess Burst Size	10	10	10

Table 7 3 Property Setting for trTCM

7.2.2 checkConforming Module Testing

The first step to do the checkConforming module test is to make sure the switch control structure can get the correct value of marker from user input.

Marker selected by user: srTCM

Output of System.out.println("srTCM>>" +sw_marker.getValue()) : 0

Marker selected by user: trTCM

Output of System.out.println("trTCM>>" +sw_marker.getValue()) : 1

As the switch case can get the correct value of marker, the algorithm for each marker will be select correctly according to the user's choice. By using the configuration in 7.1.1 the results of testing are as below:

Marker	Frame 1			Frame 2			Frame 3			Frame 4		
	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3
srTCM	-	-	-	0	0	-2	-2	-1	-1	0	0	-2
trTCM	-	-	-	0	0	0	-2	-1	0	0	0	0

Table 7 4 Results of checkConforming Module Testing

From the results shown as above, the frame have been given the conform_level value according to the colour (green=0, yellow=-1 & red=-2).

7.2.3 Module Testing in srTCM

By using the value input as shown in table 7.1, the testing results for sw_tc_perform module testing are shown as below:

	Frame.t_profiel			ProfileRecord	
	meanrate	burstsize	Burstsize2	Token_count1	Token_count2
1.	50.0	53	53	53	53
2.	55.5	60	80	60	80
3.	105.3	73	4	73	4

Table 7 5 Results of Module Testing in srTCM

The second sub test in module testing for sw_tc_perform is to ensure that the conform_level and the tc_drop have got to correct value. Below are the properties settings that will be use for the module testing:

Test 1 Result:

	Frame 1	Frame 2	Frame 3	Frame 4
Conform_level	-	0	-2	0
Tc_drop	-	False	True	False

Test 2 Result:

	Frame 1	Frame 2	Frame 3	Frame 4
Conform_level	-	0	-1	0
Tc_drop	-	False	False	False

Test 3 Result:

	Frame 1	Frame 2	Frame 3	Frame 4
Conform_level	-	-1	-2	-1
Tc_drop	-	False	True	False

From the test results in the tables at above, we set the properties so that only four frames will be sent in each tests. 1st frame send out, will not monitor by traffic conditioner. Therefore, the traffic performs result an get from 2nd frame onwards. In the tests, the bit rate of application had been set to violate the traffic profile. In test 1st test, the peak /

excess burst size are less than the packet size, therefore the 3rd frame has been marked as red and the tc_drop will be set as true. In the 2nd test, the 3rd frame has been marked as yellow colour as the peak / excess size is enough for the packet size. Lastly, in the 3rd test, the committed burst size is too small for the packet, no frame will be marked as green. The result of the conform_level is exactly same as the results shown in 7.1.2 where the values return from the checkConforming.

7.2.4 Module Testing in trTCM

The module testing in trTCM is as the test for srTCM. The configuration of the topology is shown in table 7.2. Below are the result of testing:

Test 1 Result:

	Frame 1	Frame 2	Frame 3	Frame 4
Conform_level	-	0	-2	0
Tc_drop	-	False	True	False

Test 2 Result:

	Frame 1	Frame 2	Frame 3	Frame 4
Conform_level	-	0	-1	0
Tc_drop	-	False	False	False

Test 3 Result:

	Frame 1	Frame 2	Frame 3	Frame 4
Conform_level	-	0	0	0
Tc_drop	-	False	False	False

Logically, peak rate always set greater than committed rate. In the module test for trTCM, all the peak rate has been set greater than committed rate. To test the conform_level and tc_drop in trTCM, all the token bucket size has been fixed as 10 bytes. In the 1st test, the bit rate was violate both committed rate and peak rate, therefore the 3rd frame has been marked as red and the tc_drop has been set as true. In 2nd test, the bit rate only violate the committed rate, so the 3rd frame will not mark as red but yellow colour. Finally, in the 3rd test, the bit rate neither violate the committed rate nor peak rate, in this case no frame will me marked as red or yellow.

The main purpose this test is to check whether the module has set the correct value of tc_drop according to the conform_level as set in the rules of srTCM and trTCM. From the result shown as above, this module has been developed successfully as the drop results are match with the rules.

The result of the conform_level is exactly same as the results shown in 7.1.2 where the values return from the checkConforming.

7.3 System Testing

The main purpose of simulator system testing is to test the traffic conditioner marker. Marker testing is carried out with the purpose to make sure that the srTCM and trTCM in traffic conditioner is function correctly and successfully in UMJanetSim.

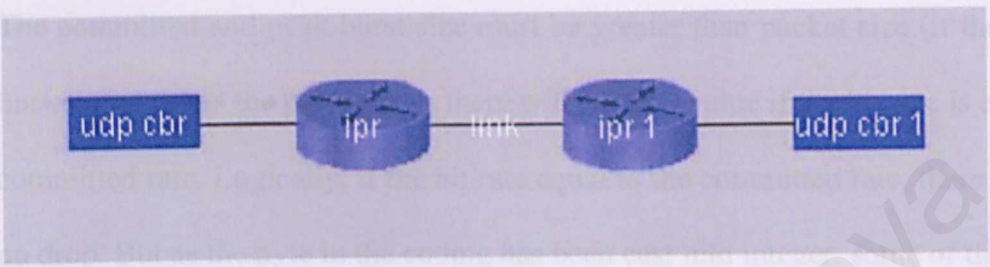


Figure 7 2 Topology for System Testing

The basic properties setting for system testing in above topology are shown in table 7.6.

Property	Value
Packet Size (udp cbr)	53
Number of Mbits to be sent (udp cbr)	0.424
Destination IP (udp cbr)	10.0.0.2
IP address to link (ipr)	10.0.0.1
IP address to link (ipr 1)	10.0.0.2

Table 7 6 Default property Setting for System Setting

In the system testing, the source is sent from udp cbr to ipr 1. In this case, ipr is the edge router of the network, therefore it will handle the traffic conditioner role for this system testing. This is the simplest testing where it is one-way transmission in this topology.

7.3.1 System Testing for srTCM

In the srTCM system testing, the values that need to be chosen are very important. There are few rules that must be following to test the functionality of srTCM:

I. Bit rate in UDP_CBR must be less than link speed (to avoid queue dropped)

II. The committed and peak burst size must be greater than packet size (if the token bucket size set as the packet size, there will be drop value if the bit rate is equal to committed rate. Logically, if the bit rate equal to the committed rate, there will be no drop. But as the byte in the coding has been cast into integer, some of the value will be smaller for example when the value of 52.985 cast into integer, it will become 52 where it is smaller than the packet size and will cause frame dropped in the test case.)

From the results shown is table 7.7, I conclude that:

I. If the bit rate does not violate the committed rate, all the frames will be sent to the destination router.

II. If the bit rate violates the committed rate, parts of the frames (marked as red) will be dropped. Besides, some of the frame will be marked as yellow colour.

III. Test 3 shows that, if the bandwidth is busy (link speed set as 80 to fully utilize the queue) some of the frames marked as yellow will be dropped.

IV. Test 5 shows that, if bit rate equal to the committed rate does not mean that no frames will be dropped. The casting of token count from double to integer cause it less 1 token to allow the frame to be marked as green colour.

	Test 1	Test 2	Test 3	Test 4	Test 5
Bit rate	80.0	80.0	80.0	80.0	90.0
Total frame sent	1000	1000	1000	1000	1000
Total frame dropped by TC	0	332	336	285	249
Committed rate	80.0	60.0	60.0	60.0	90.0
Committed burst size	54	54	54	60	53
Peak/excess burst size	54	54	54	60	53
Frames dropped %	0	33.1668	33.4661	28.4715	24.8751
Frames marked green	999	500	500	500	500
Frames marked yellow	0	167	167	214	250
Frames marked yellow (dropped by TC)	0	0	4	0	0
Frames marked red	0	332	332	285	249
Link speed	500	500	80	500	500
Frames received	1001	669	665	716	752

Table 7 7 Results of System Testing in srTCM

7.3.2 System Testing for trTCM

System testing of trTCM is almost the same like the system testing for trTCM, the only different for trTCM is in trTCM, peak rate value haa to input by user. In trTCM testing, the peak rate has to be greater than committed rate, as been explained in the theory of trTCM.

	Test 1	Test 2	Test 3	Test 4
Bit rate	40.0	55.0	80.0	80.0
Total frame sent	1000	1000	1000	1000
Total frame dropped by TC	0	0	263	0
Peak rate	60.0	60.0	60.0	60.0
Committed rate	50.0	50.0	50.0	50.0
Committed burst size	54	54	54	54
Peak/excess burst size	54	54	54	54
Frames dropped %	0	0	26.2737	34.0637
Frames marked green	999	905	618	618
Frames marked yellow	0	94	118	118
Frames marked yellow (dropped by TC)	0	0	0	79
Frames marked red	0	0	263	263
Link speed	500	500	500	80
Frames received	1001	1001	738	662

Table 7 8 Results of System Testing in trTCM

From the results shown is table 7.8, I conclude that:

I. If the bit rate does not violate the committed rate, all the frames will be sent to the destination router
II. If the bit rate violates the committed rate, parts of the frames will be marked as yellow colour. The yellow frame will only be drop when the queue of output is nearly full.
III. If the bit rate violates both committed rate and peak rate, the will be frames dropped happened.

7.3.3 System Testing for Other Toplogies

Besides the testing showing at above, there are also system testing done on different topologies. Different test cases have been created to show the availability of three colour marker. Please refer to the appendixes for references. The purposes of these testing are:

I. To ensure that the three colour marker can also be apply on more than one application sender.
II. To differentiate the result of individual testing and global testing
III. To have a test on two way transmission
IV. To test that the system can be run with routers with different type of three colour marker.

7.4 Summary

This chapter gives an idea hot to test the traffic conditioner in UMJanetsim. In this testing, Testing need to be done from component, module and lastly is system testing. The method of bottom-up testing is easy to trace the fault or mistake. It allow developer to solve the bugs in the bottom layer before go into the top layer. In the testing, bit rate, committed rate and peak rate is the major parameters. Rates are the factors to cause the frame drop but the burst size is only affect the percentage of drop.

CHAPTER 8 CONCLUTION

In network simulator, there are numerous traffic conditioners to monitor the traffic of network and each has its own merits and drawbacks. This report presents three colour marker for traffic conditioner performing in UMJanetSim using object-oriented programming.

The use of Object-Oriented Programming approach has provided a several key benefits to the development of the network simulator.

The main advantage of object-oriented programming techniques is its modularity. It allows the developers to create new modules that do not need to be modify when a new type of object is added. Every object forms a separate entity whose internal workings are decoupled from other parts of the system. The use of this approach provides the network simulator with other features such as simplicity, modularity, extensibility, maintainability and reusability.

Java is an object oriented language with the benefits of built in support for multithreading and the ability of threads to run simultaneously. The object-oriented approach is one of the requirements to complete the development of this project. Finally, cross platforms ability of Java, fulfill the objectives of this project too.

CHAPTER 8 CONCLUSION

In network simulator, there are numerous traffic conditioners to monitor the traffic of network and each has its own merits and drawbacks. This report presents three colour marker for traffic conditioner performing in UMJanetSim using object-oriented programming.

The use of Object-Oriented Programming approach has provided a several key benefits to the development of the network simulator.

The main advantage of object-oriented programming techniques is its modularity. It allows the developers to create new modules that do not need to be modify when a new type of object is added. Every object forms a separate entity whose internal workings are decoupled from other parts of the system. The use of this approach provides the network simulator with other features such as simplicity, modularity, extensibility, maintainability and reusability.

Java is an object oriented language with the benefits of built in support for multithreading and the ability of threads to run simultaneously. The object-oriented approach is one of the requirements to complete the development of this project. Finally, cross platforms ability of Java, fulfill the objectives of this project too.

This project is a simulation of diffserv network with traffic conditioner. The existing UMJanetSim is based on the object-oriented approach where the classes with corresponding attributes and functions are defined first. Three colour marker modules are designed and implemented based on the object-oriented approach as in the existing simulator.

In the implementation stage, modules in Java code with the suitable algorithm are added in UMJanetSim 0.66. At the end of the implementation, testing is done on all of the Java classes to ensure the integrated modules work perfectly. The testing phase is to find and solve the error sso that the implementation of three colour colour will achieve the objective of this project.

Finally, the complete set of simulator with traffic conditioner is ready. In other word, the project objectives, goals and scopes are achieved. The following sections highlights simulator strengths, limitations as well as the proposed future enhancements.

8.1 System Strengths.

UMjanetsim is developed wit its strengths, which are:

- **Object-oriented:** UMjanetSim is fully object-oriented. All the functions and modules are built in class. The additional functions or modification can be done easily.
- **User friendly:** The traffic conditioner is added as one of the function which user is allow to choose it by tick the checkbox in router's or application's property. The design of UMJanetSim is simple and easy to use.
- **Availability:** User is allowed to choose both srTCM and trTCM in a single topology by set it in two routers respectively. The availability to choose both type of three color marker allow user to compare both three colour marker's result in one test case. Besides that, there are tow type of traffic profile testing. The first one is to set it at application which is for individual testing purpose. Next is a global setting in router, which is for all the applications connected to that particular router.

8.2 System Limitations

- **None web-based system:** UMJanetSim worked based on Java application. It is only able to work in platform with java environment. Therefore, it is not a web-enabled system.
- **UDP_CBR:** The development of three colour marker is done for UDP_CBR application. Other type of applications can not simulate with

three colour marker traffic conditioner although the traffic conditioner in router is enabled.

- **CPU speed:** UMjanetSim needs a high speed CPU in order to run smoothly.

8.3 *Future Enhancement*

- The implementation of three colour marker should include in other application in UMJanetSim.
- The graphic interface should be improved so that the frame transmission can show graphically on the screen to allow user analyses and monitor the traffic pattern without studying on the value.
- A graph to provide number of frames marked as different colour should be prepared so that it is easier to compare three types of frames

REFERENCES

Cisco Systems. 2001. DiffServ—The Scalable End-to-End QoS Model [online]. Cisco Systems, Inc. Available from: www.cisco.com/warp/public/cc/pd/iosw/ioft/iofwft/prodlit/difse_wp.pdf [Accessed 25 June 2003]

Cisco Systems. Cisco − Implementing Quality of Service Policies with DSCP [online] Available from: www.cisco.com/warp/public/105/dscpvalues.html [Accessed 30 June 2003]

Deitel. 1997. C++ How to Program. New Jersey: Prentice Hall

Deitel. 1997. Java How to Program. New Jersey: Prentice Hall

Dr. Ljiljana Trajkovic. 2003. IP Marking, Metering, and Management [online]. Available from: ttp://www.ensc.sfu.ca/~ljilja/ENSC833/Projects/uy_xu/ensc833_ax_ju.pdf [Accessed 3 July 2003]

Evi Tsolakou, I. S. Venieris. Implementation of Traffic Conditioning and PHB mechanisms in OPNET [online]. Available from: www-st.inf.tu-

dresden.de/aquila/files/ pub/comcon8-ntu-opnet-paper.pdf [Accessed 23 July 2003]

Forouzan, Behrouz A. 2000 Data communications and networking. Boston: McGraw-Hill

Ilias Andrikopoulos, Lloyd Wood and George Pavlou Centre . A Fair Traffic Conditioner for the Assured Service in a Differentiated Services Internet.[online] Available from: www.gta.ufrj.br/diffserv/Andrik-00a.pdf [Accessed 25 June 2003]

Jerry Ryan. 1998. Multiprotocol Label Switching (MPLS). [online]. Available from: http://www.itpapers.com/abstract.aspx?scid=98&docid=8055 [Accessed 12 July 2003]

Lynda Linney. 1999. Differentiated Service on IBM 221 x Routers [online]. Available from: http://www.cnaf.infn.it/~ferrari/tfng/ds/ibmtest/diff_serv-handouts.pdf [Acessed 5 July 2003]

M. Melly. 1995. Object Oriented Basic Concepts and Advantages [online]. Availale from: www.mmrg.ecs.soton.ac.uk/publications/archieve/melly1995a/html/ node3.html [Accessed 15 August 2003]

Mohammed Atiquzzaman. 2001. ItswTCM: A New Aggregate Marker to Improve Fairness in DiffServ. [online] Available from: citeseer.nj.nec.com/su01itswtcm.html [Accessed 18 July 2003]

Muhammad Jaseemuddin. A Study of Profiled Handoff for DiffServ-Based Mobile Nodes. [online] Available from: www.ee.ryerson.ca/~jaseem/pub/wj_2002.pdf

Nicolas Christin. 2000. Current Directions in the DiffServ World [online]. Available from: www.cs.virginia.edu/~nc2y/talks/diffserv-2000.pdf [Accessed 5 July 2003]

Patrice Bellagamba. 2002. Multi-Protocol Label Switching. [online] Available from: www.cisco.com/global/FR/documents/pdfs/sp/getronics_0302/ MPLS_Technical_Introduction.pdf [Acessed 10 July 2003]

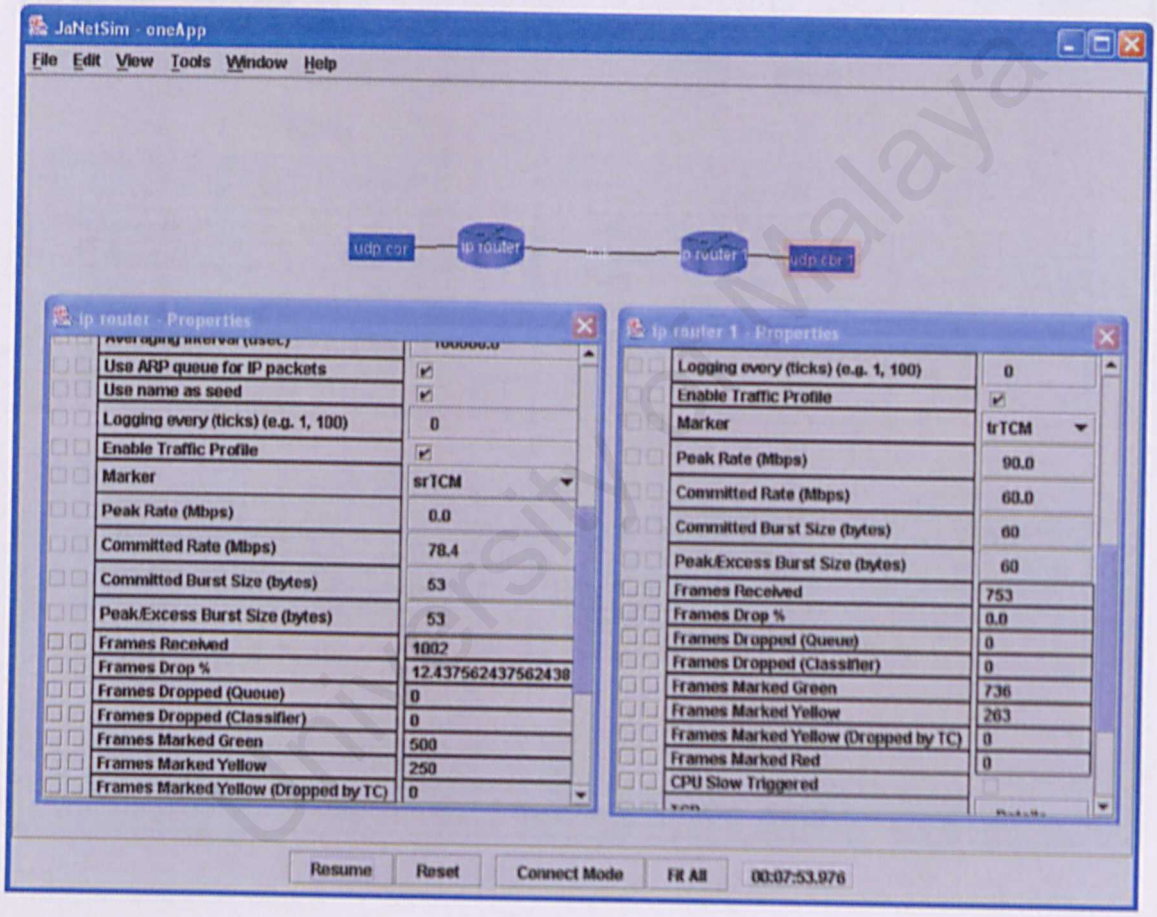
Richard Forberg. 2001. Internet Protocol Quality of Service (IP QoS) Using DiffServ for Application-Specific Service Level Agreements. [online]. Available from: www.quarrytech.com/IP_QoS_Whitepaper.pdf [Accessed 5 July 2003]

Youngtak Kim. 2002. DiffServ-aware-MPLS Networking: a Promising Traffic Engineering for Next Generation Internet (NGI) [online]. Available from: http://dpnm.postech.ac.kr/conf/apnoms2002/tutorial/Tutorial-YTKim.pdf [Accessed 20 July 2003]

J. Heinanen and R.Guerin. 1999. A Single Rate Three Color Marker. University of Pennsylvania.

J. Heinanen and R.Guerin. 1999. A Two Rate Three Color Marker. University of Pennsylvania.

APPENDIXES


JaNetSim - oneApp
File Edit View Tools Window Help
udp cor
ip router
p router
udp cbr 1
ip router - Properties
Use ARP queue for IP packets
Use name as seed
Logging every (ticks) (e.g. 1, 100) 0
Enable Traffic Profile
Marker srTCM
Peak Rate (Mbps) 0.0
Committed Rate (Mbps) 78.4
Committed Burst Size (bytes) 53
Peak/Excess Burst Size (bytes) 53
Frames Received 1002
Frames Drop % 12.437562437562438
Frames Dropped (Queue) 0
Frames Dropped (Classifier) 0
Frames Marked Green 500
Frames Marked Yellow 250
Frames Marked Yellow (Dropped by TC) 0
ip router 1 - Properties
Logging every (ticks) (e.g. 1, 100) 0
Enable Traffic Profile
Marker trTCM
Peak Rate (Mbps) 90.0
Committed Rate (Mbps) 60.0
Committed Burst Size (bytes) 60
Peak/Excess Burst Size (bytes) 60
Frames Received 753
Frames Drop % 0.0
Frames Dropped (Queue) 0
Frames Dropped (Classifier) 0
Frames Marked Green 736
Frames Marked Yellow 263
Frames Marked Yellow (Dropped by TC) 0
Frames Marked Red 0
CPU Slow Triggered
Resume
Reset
Connect Mode
Fit All
00:07:53.976